

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

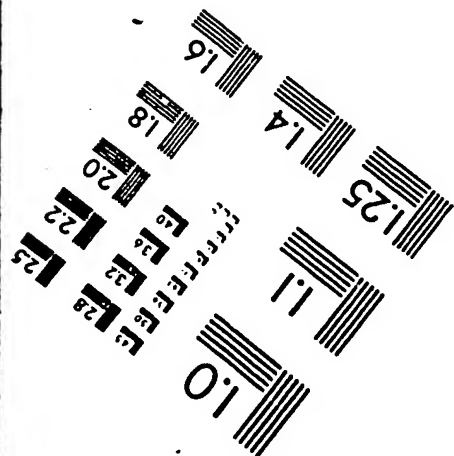
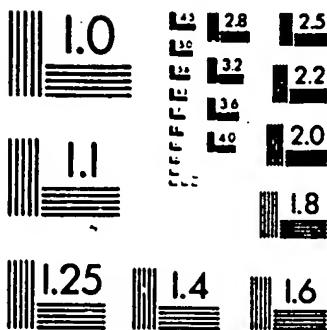
Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

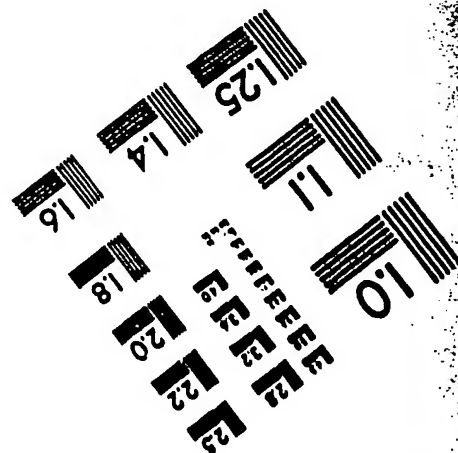
- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**



PHOTOGRAPHIC SCIENCES CORPOKATION
770 BASKET ROAD
P.O. BOX 338
WEBSTER, NEW YORK 14580
(716) 265-1600



microunity

Zeus System Architecture

COPYRIGHT 1998 MICROUNITY SYSTEMS ENGINEERING, INC. ALL RIGHTS RESERVED.



MicroUnity

Craig Hansen
Chief Architect

MicroUnity Systems Engineering, Inc.
475 Potrero Avenue
Sunnyvale, CA 94086-4118
Phone: 408.734.8100
Fax: 408.734.8136
email: craig@microunity.com
<http://www.microunity.com>

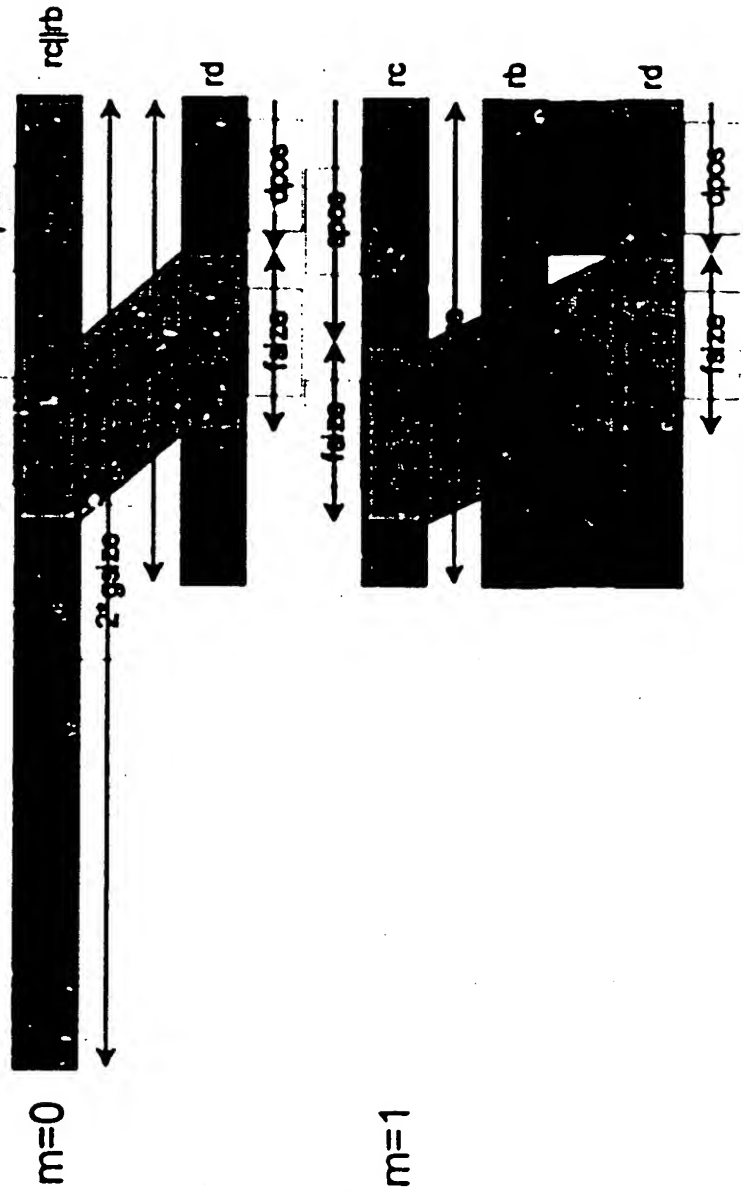


Crossbar extract control

■ layout

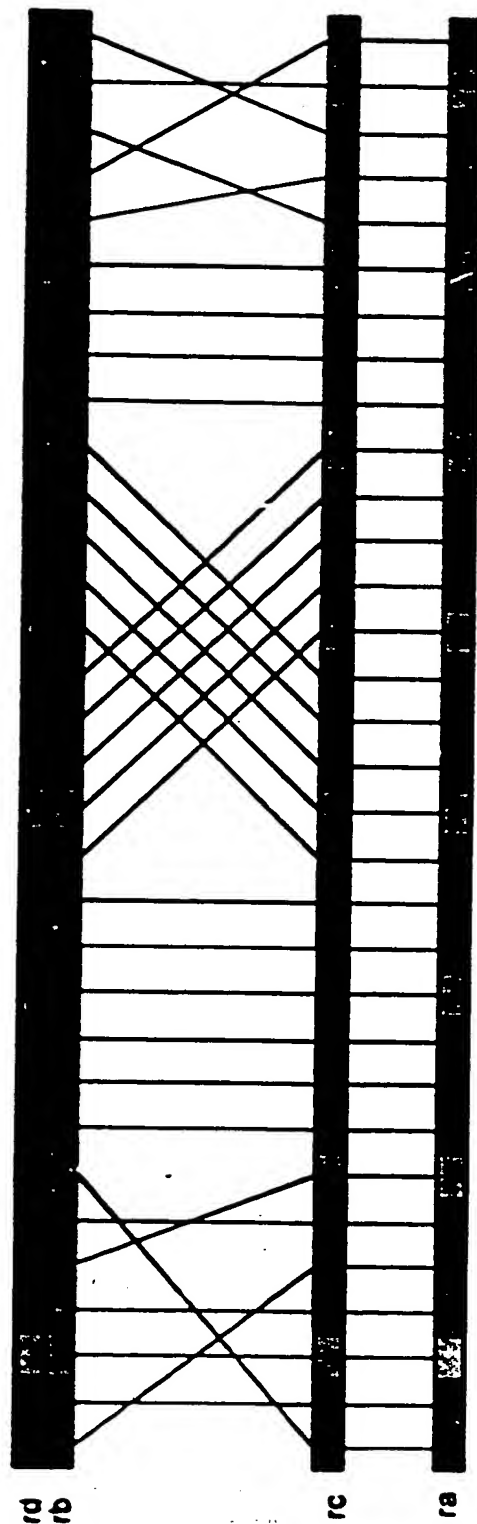


■ function



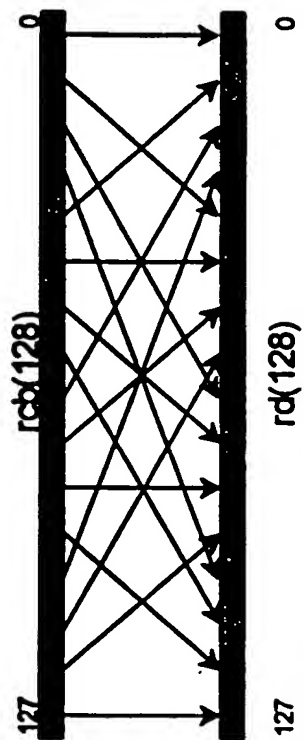
Crossbar Select bytes

- X.SELECT.8 ra=rc,rd,rb



4-way shuffle bytes within hexlet

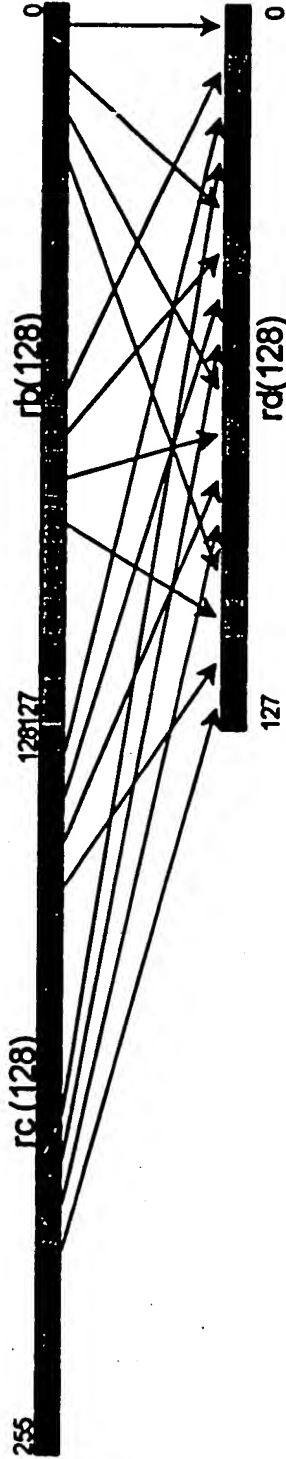
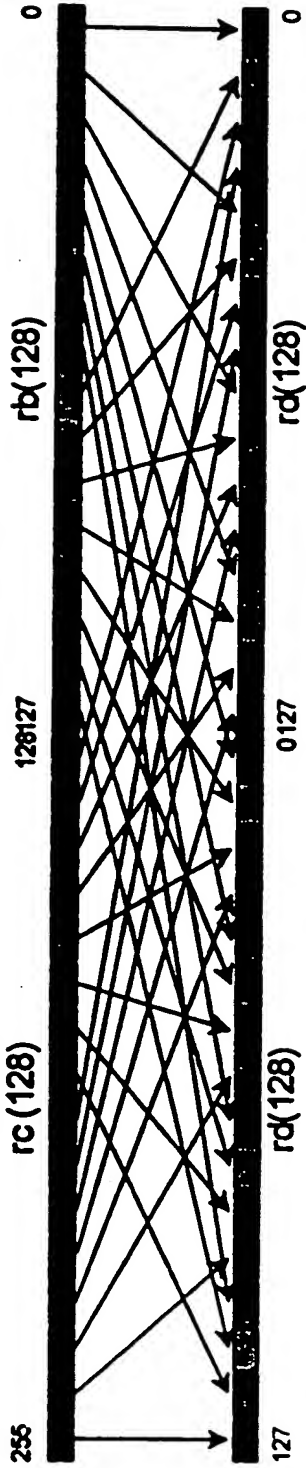
■ XSHUFFLEI.128 rd=rcb,8,4





4-way shuffle bytes within trilet

■ XSHUFFLEI.128 rd=rc,rb,8,4





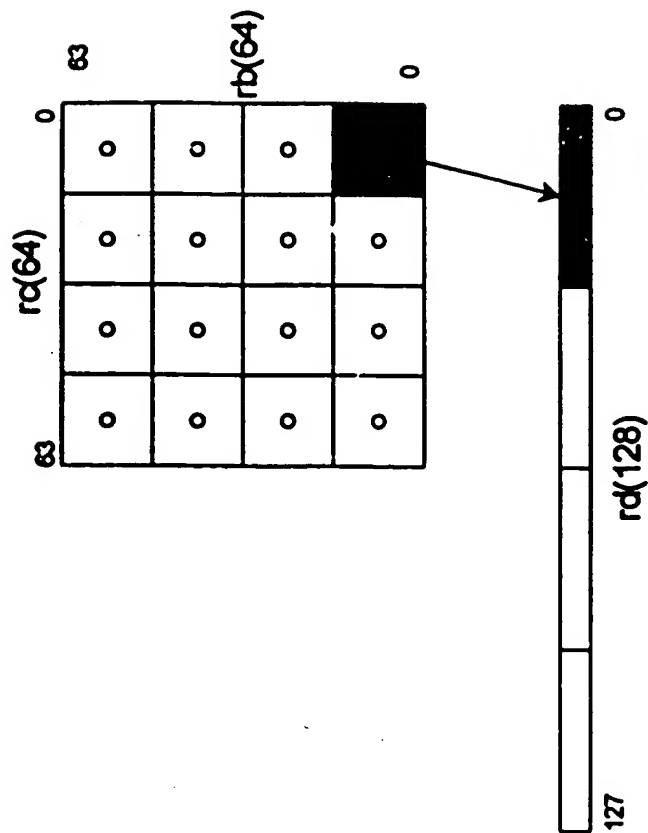
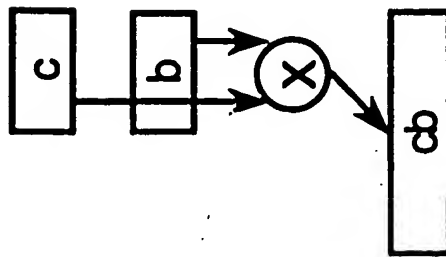
Ensemble Instructions

- Multiply
 - ◆ Fixed-point
 - size-doubling
 - extract
 - ◆ Floating-point
 - ◆ Complex
 - ◆ Polynomial
 - ◆ Galois Field
- Floating-point
 - ◆ Add, Subtract, Divide, Sum
 - ◆ Inflate, Deflate, Float, Sink
 - ◆ Reciprocal Estimate
 - ◆ Reciprocal Square Root Estimate
- Fixed-point
 - ◆ Sum
 - ◆ Log-most
- ◆ Convolve
- ◆ Multiply-add
- ◆ Scale-add
- ◆ Multiply-sum



Multiply

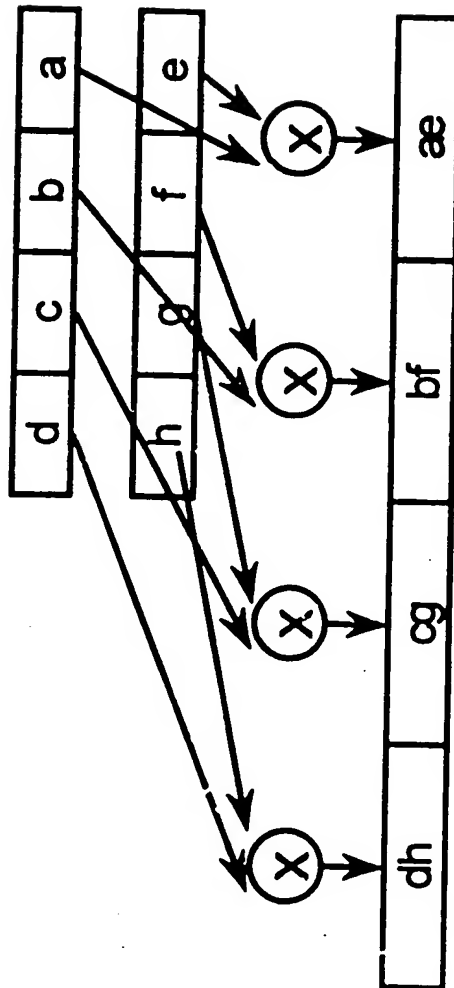
$$\blacksquare rd_{32} = rc_{16} * rb_{16}$$





Ensemble multiply

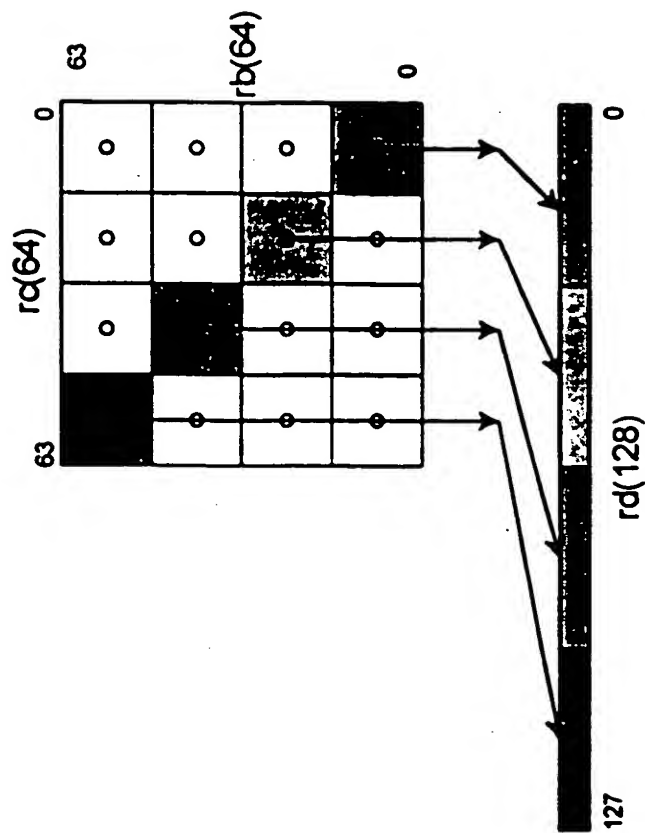
$$\blacksquare rd_{128} = rc_{64} * rb_{64}$$





Ensemble multiply

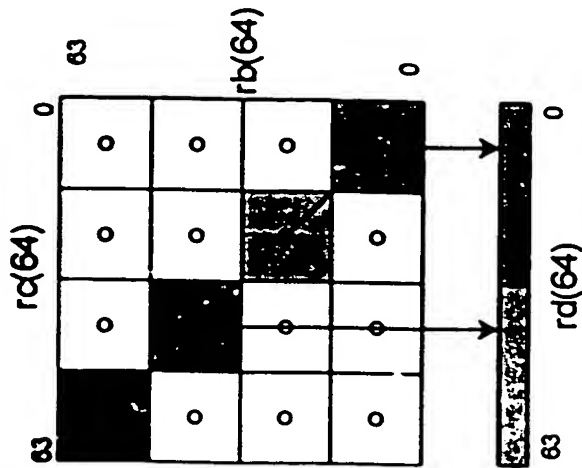
$$\blacksquare rd_{128} = rc_{64} * rb_{64}$$





MMX PMADDWD

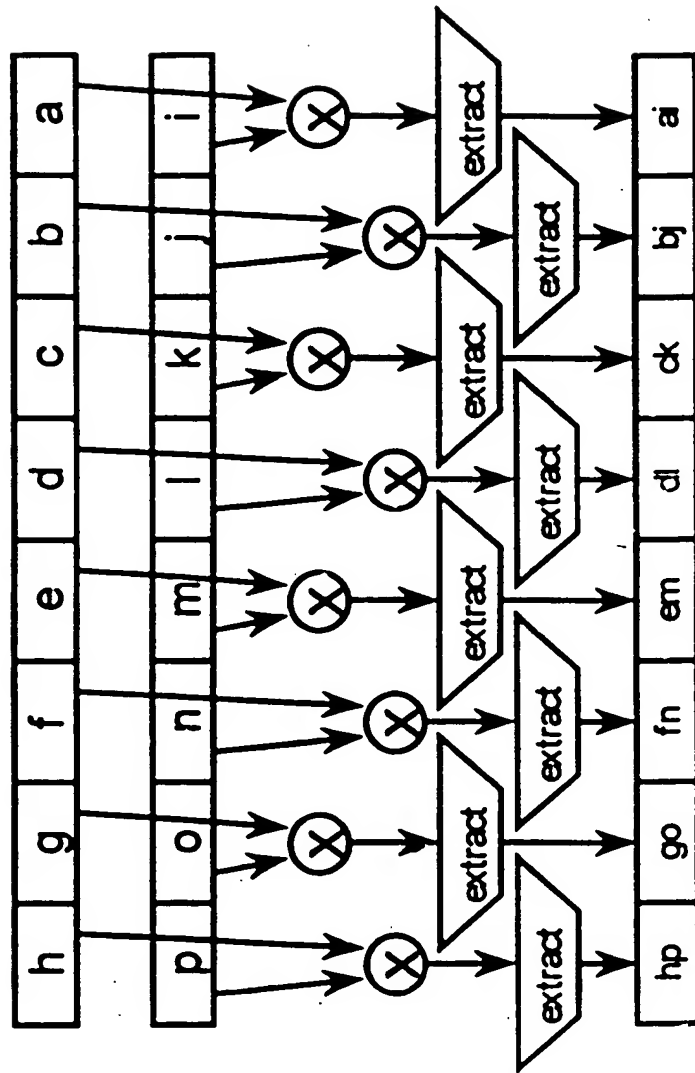
$$\blacksquare \text{rd}_{128} = \text{rc}_{64} * \text{rb}_{64}$$





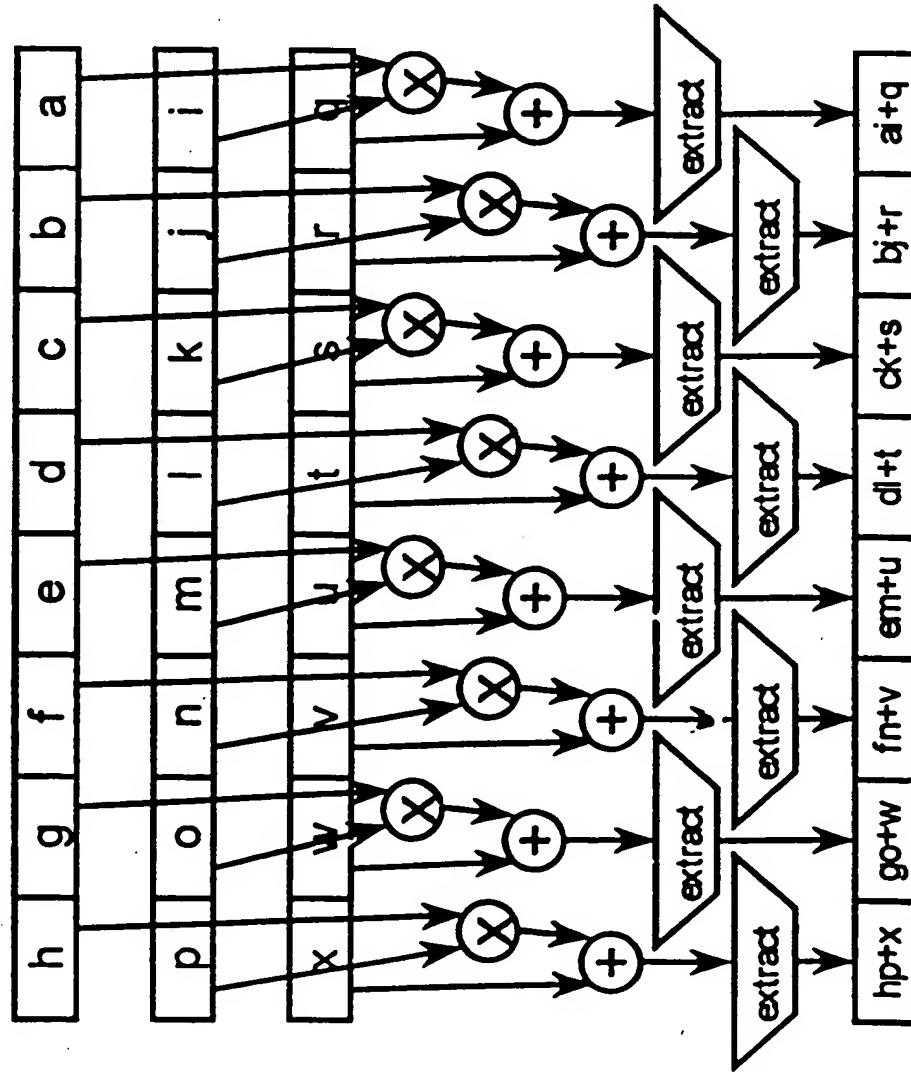
Ensemble multiply extract

$$\blacksquare rd_{128} = rc_{128} * rb_{128}$$



Ensemble multiply add extract

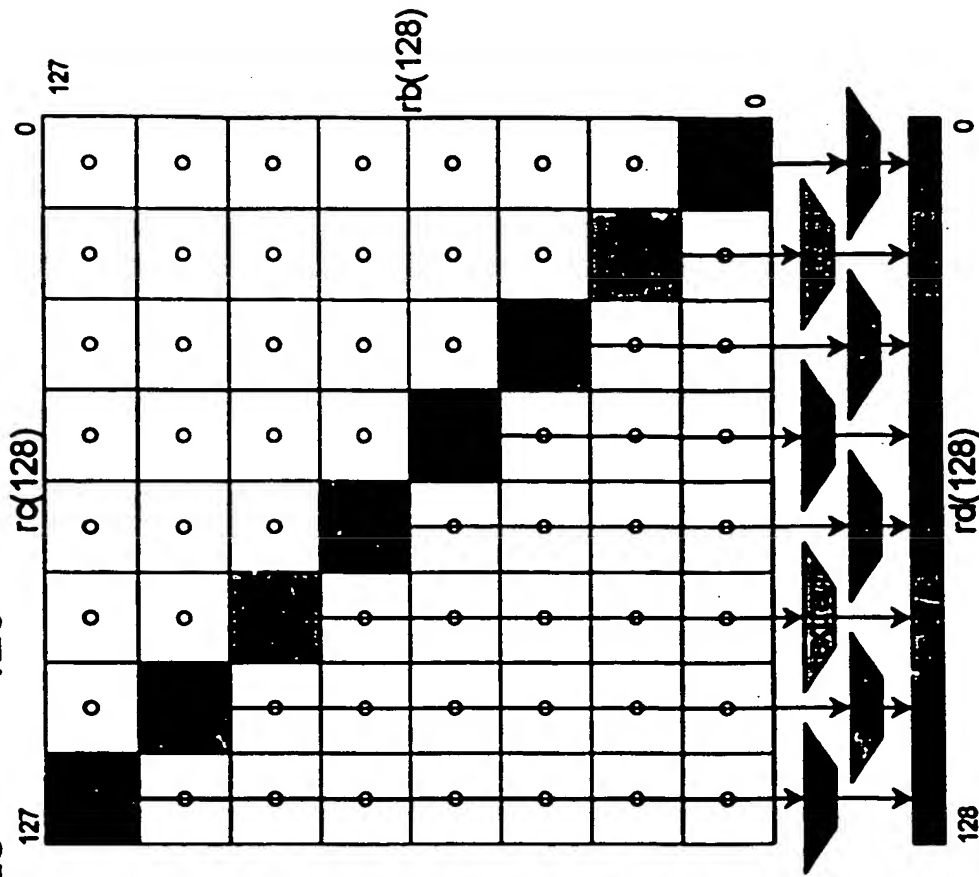
$$\blacksquare rd_{128} = rc_{128} * rb_{128} + rd_{128}$$





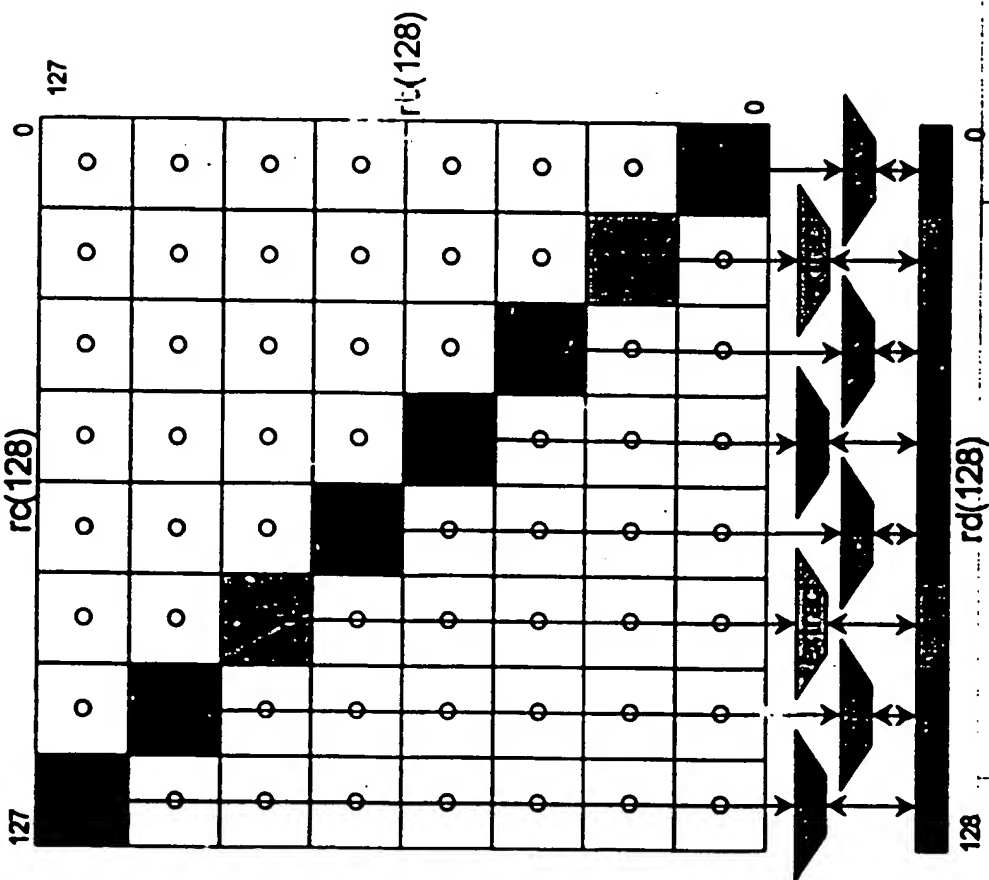
Ensemble multiply extract

$$\blacksquare \text{ rd}_{128} = \text{rc}_{128} * \text{rb}_{128}$$



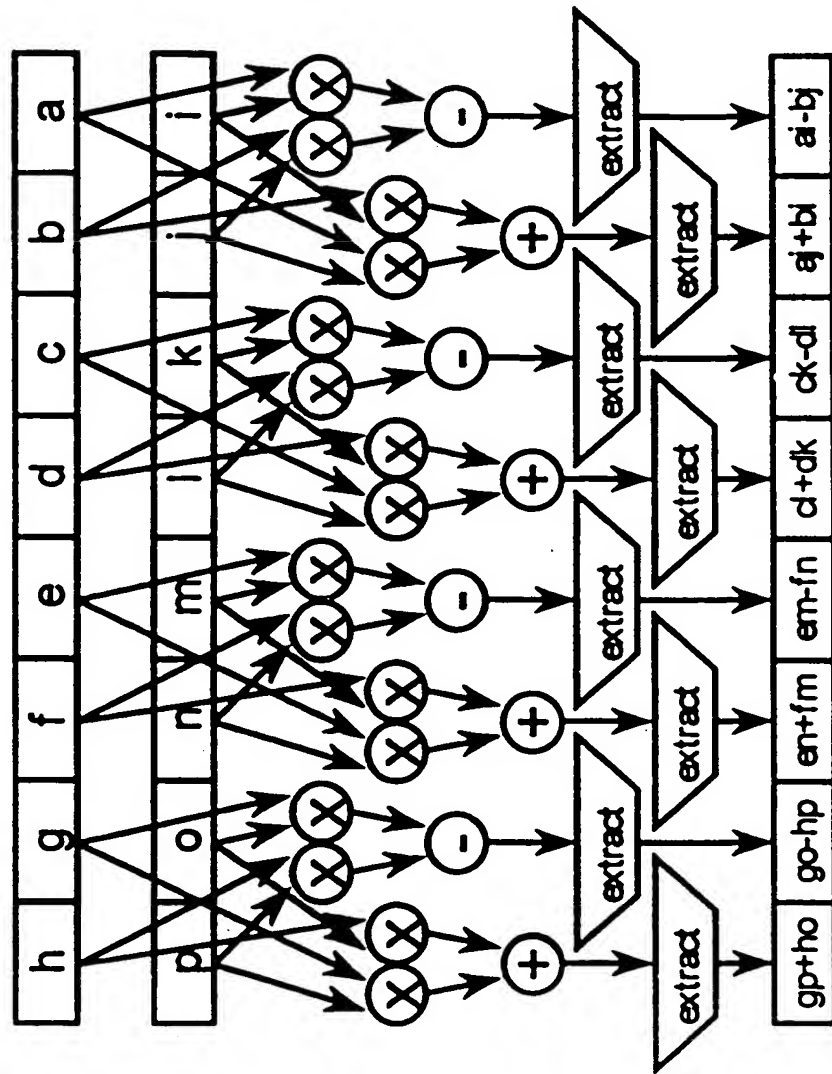
Ensemble multiply add extract

$$\blacksquare rd_{128} = rc_{128} * rb_{128} + rd_{128}$$



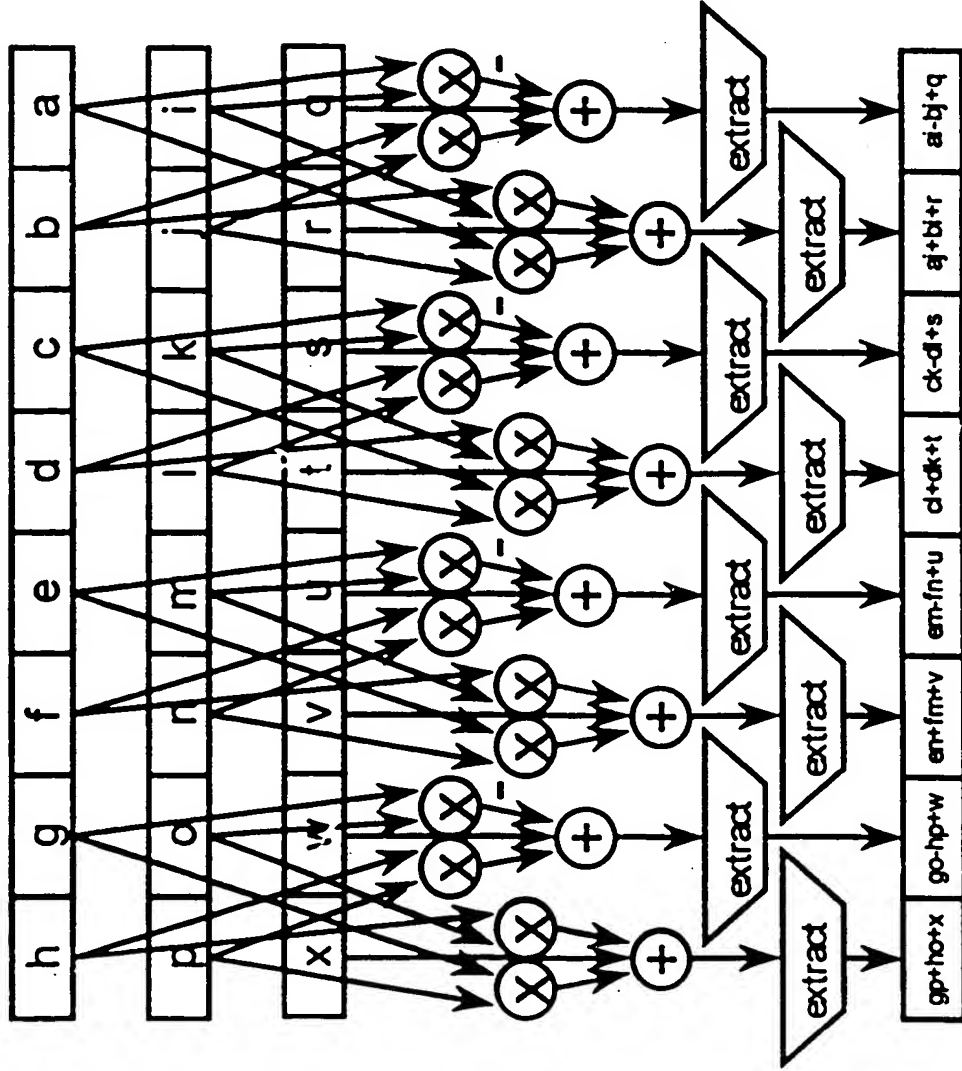
Ensemble multiply extract complex

$$\blacksquare \text{rd}_{128} = \text{rc}_{128} * \text{rb}_{128}$$



Ensemble multiply add extract complex

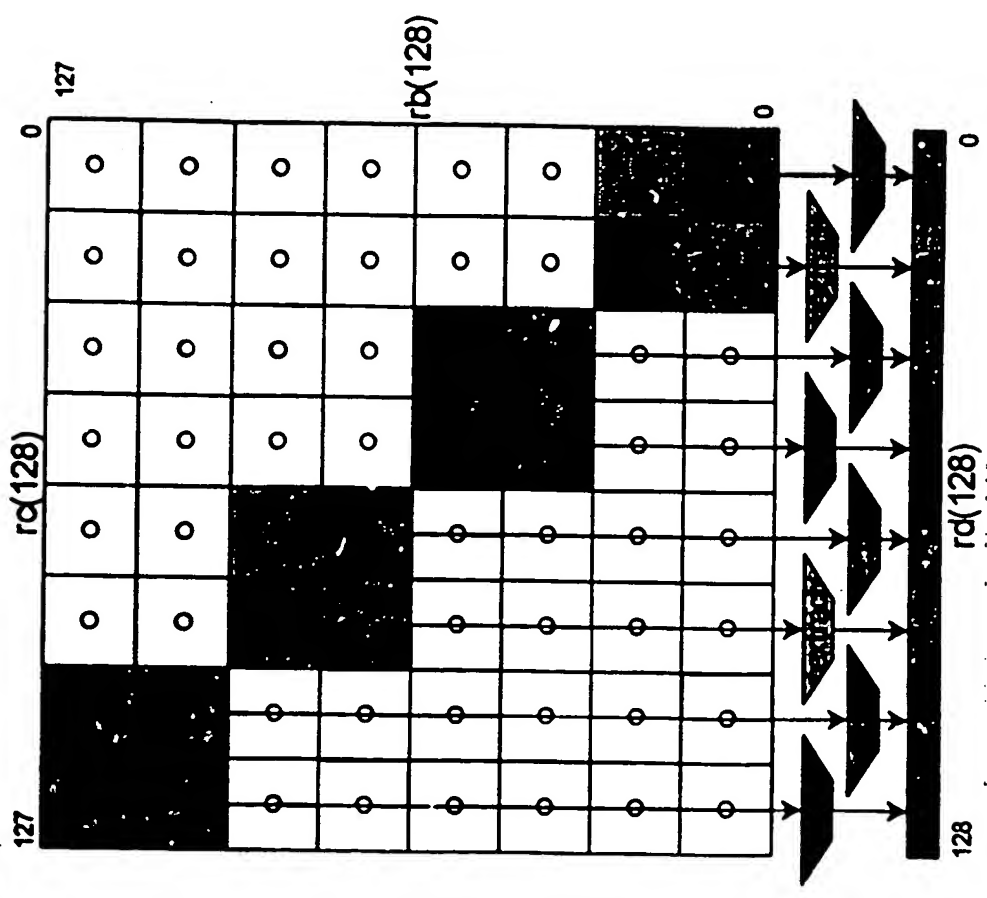
$$\blacksquare rd_{128} = rc_{128} * rb_{128} + rd_{128}$$





Ensemble multiply extract complex

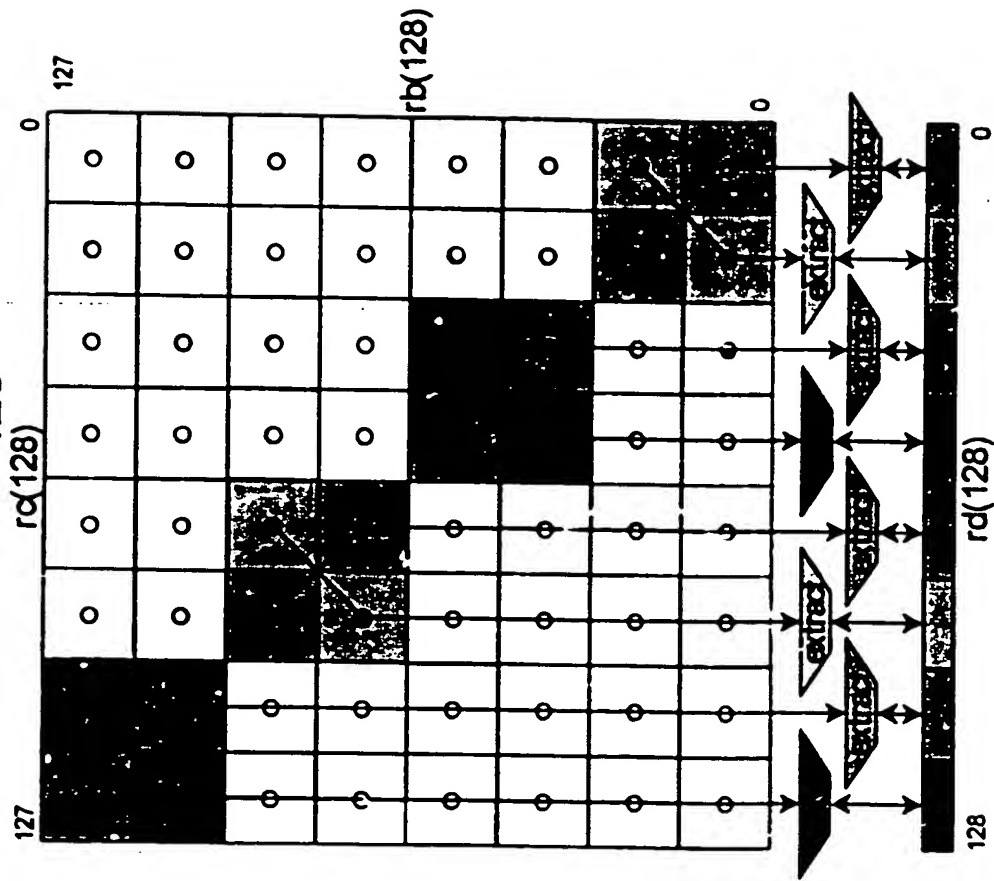
■ $rd_{128} = rc_{128} * rb_{128}$





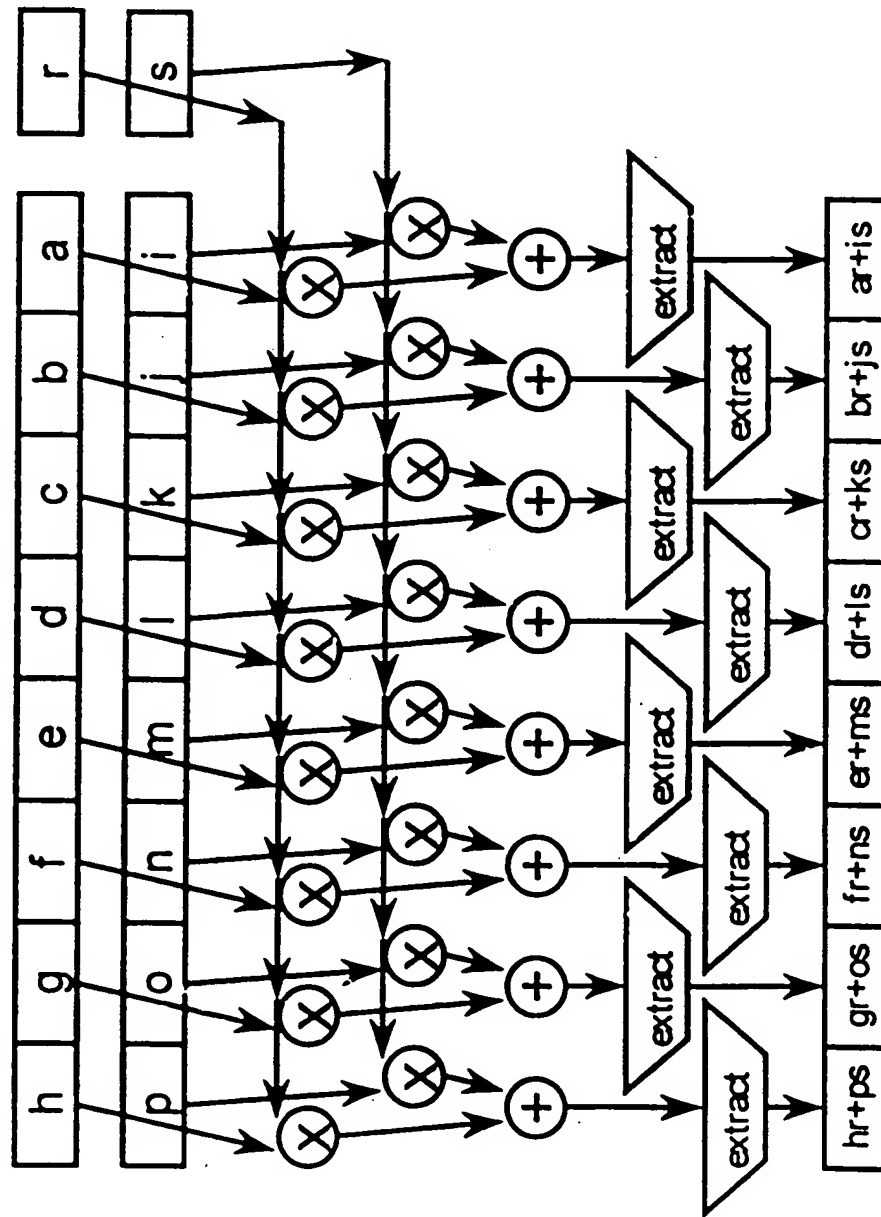
Ensemble multiply add extract complex

$$\blacksquare \text{rd}_{128} = \text{rc}_{128} * \text{rb}_{128} + \text{rd}_{128}$$



Ensemble scale add extract

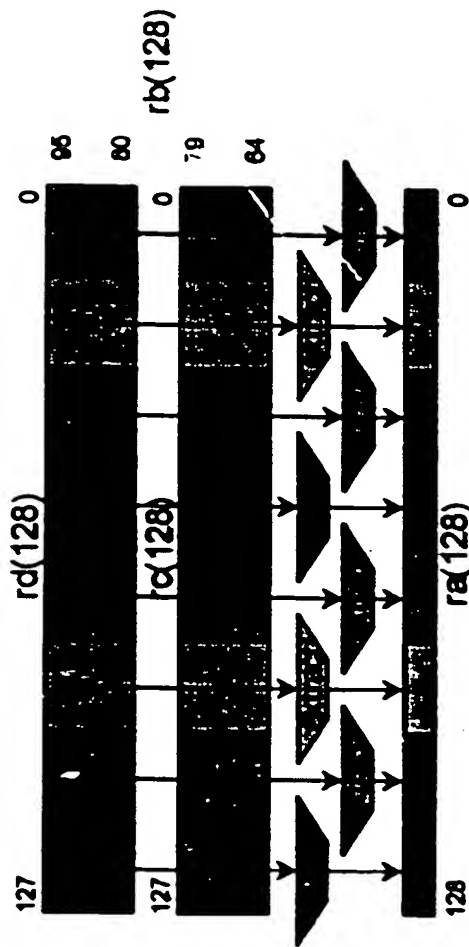
$$\mathbf{ra}_{128} = \mathbf{rd}_{128} * \mathbf{rb}_{\text{size}} + \mathbf{rc}_{128} * \mathbf{rb}_{\text{size}}$$





Ensemble scale add extract

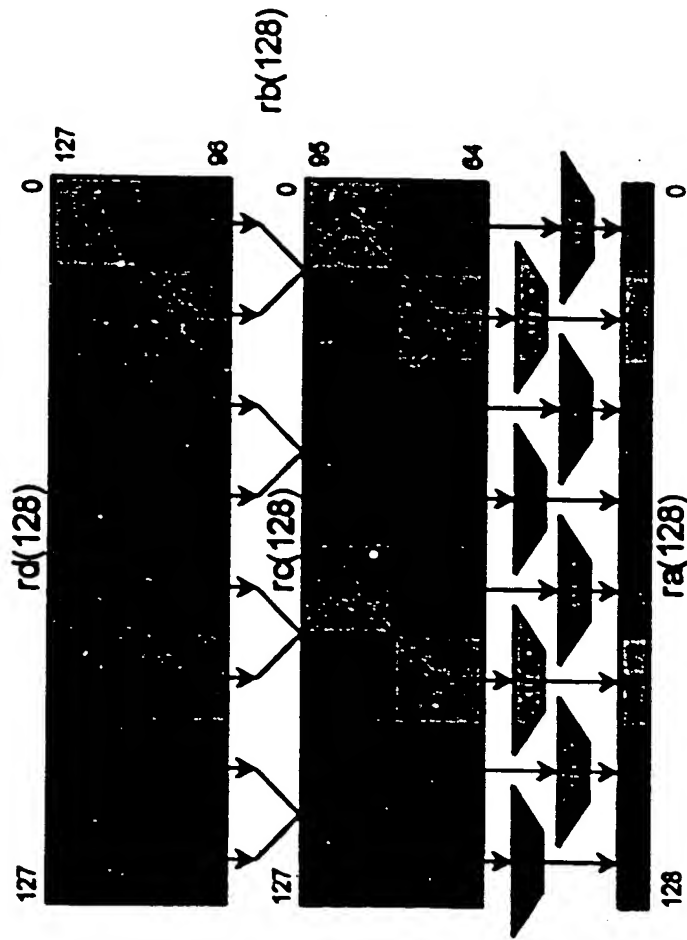
$$\blacksquare \text{ra}_{128} = \text{rd}_{128} * \text{rb}_{\text{size}} + \text{rc}_{128} * \text{rb}_{\text{size}}$$





Ensemble scale add extract complex

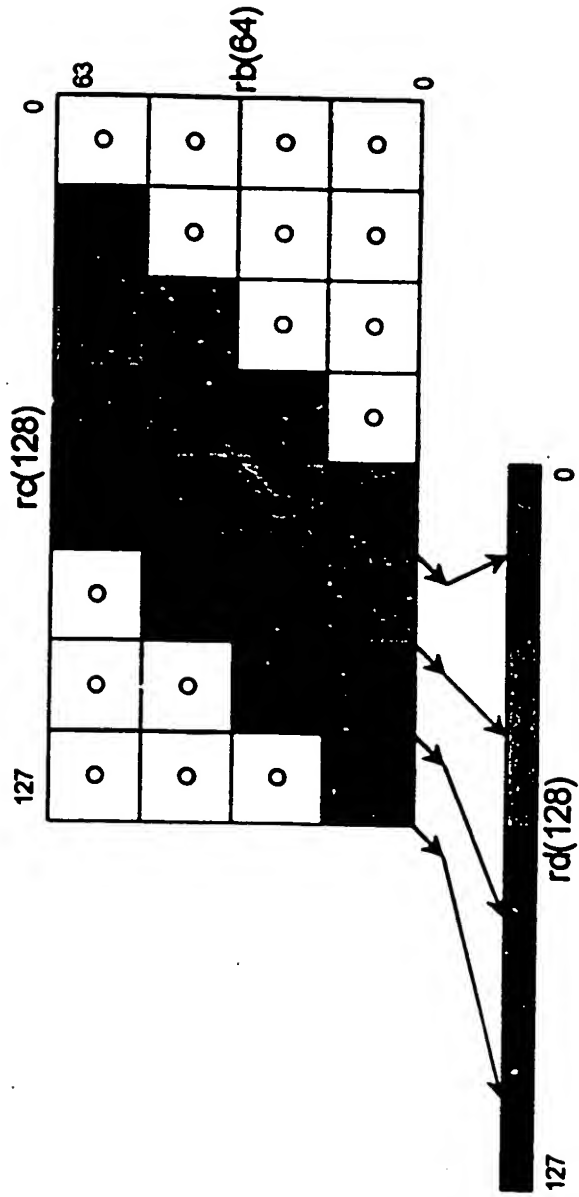
$$\blacksquare ra_{128} = rd_{128} * rb_{size*2} + rc_{128} * rb_{size*2}$$





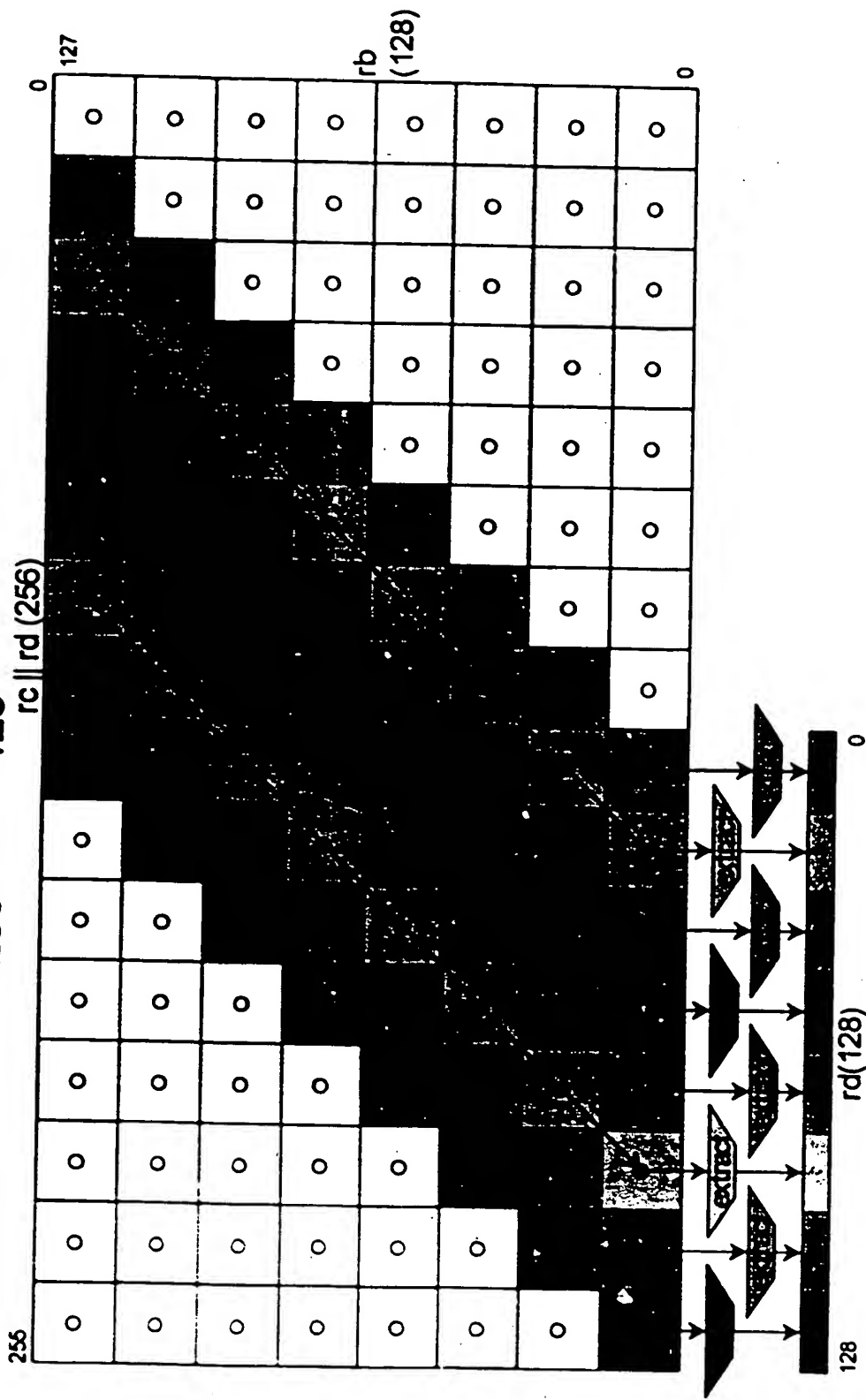
Ensemble convolve

$$\blacksquare rd_{128} = rc_{128} * rb_{64}$$



Ensemble convolve extract

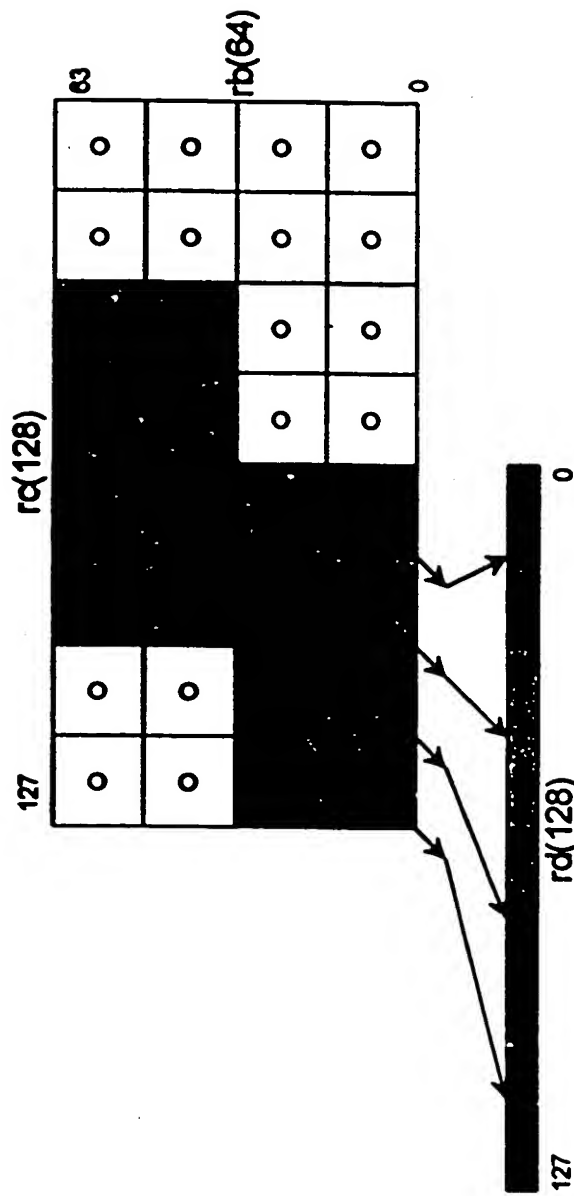
$$\blacksquare \text{rd}_{128} = (\text{rd} \parallel \text{rc})_{256} * \text{rb}_{128}$$





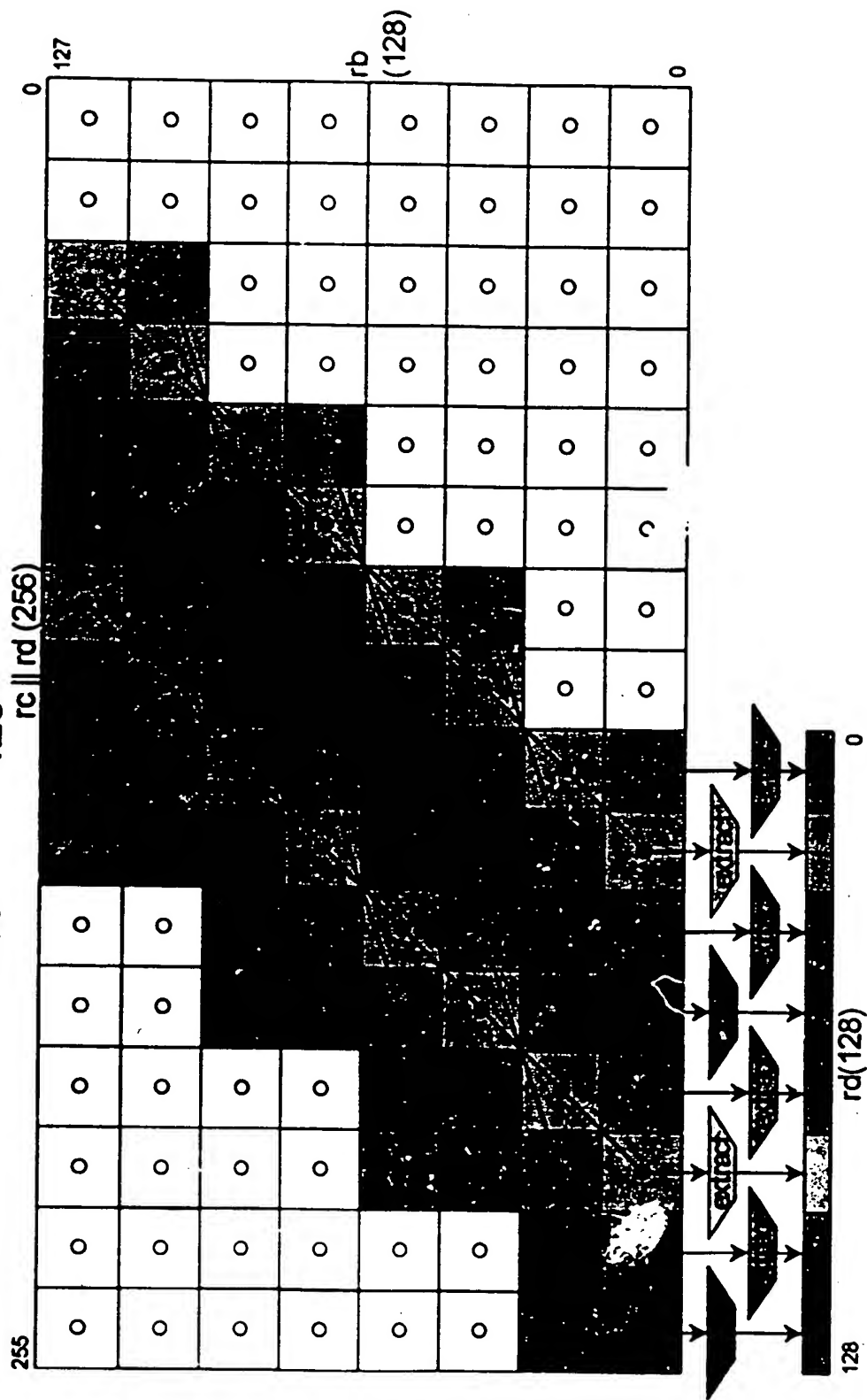
Ensemble convolve complex

$$\blacksquare \text{rd}_{128} = \text{rc}_{128} * \text{rb}_{64}$$



Ensemble convolve extract complex

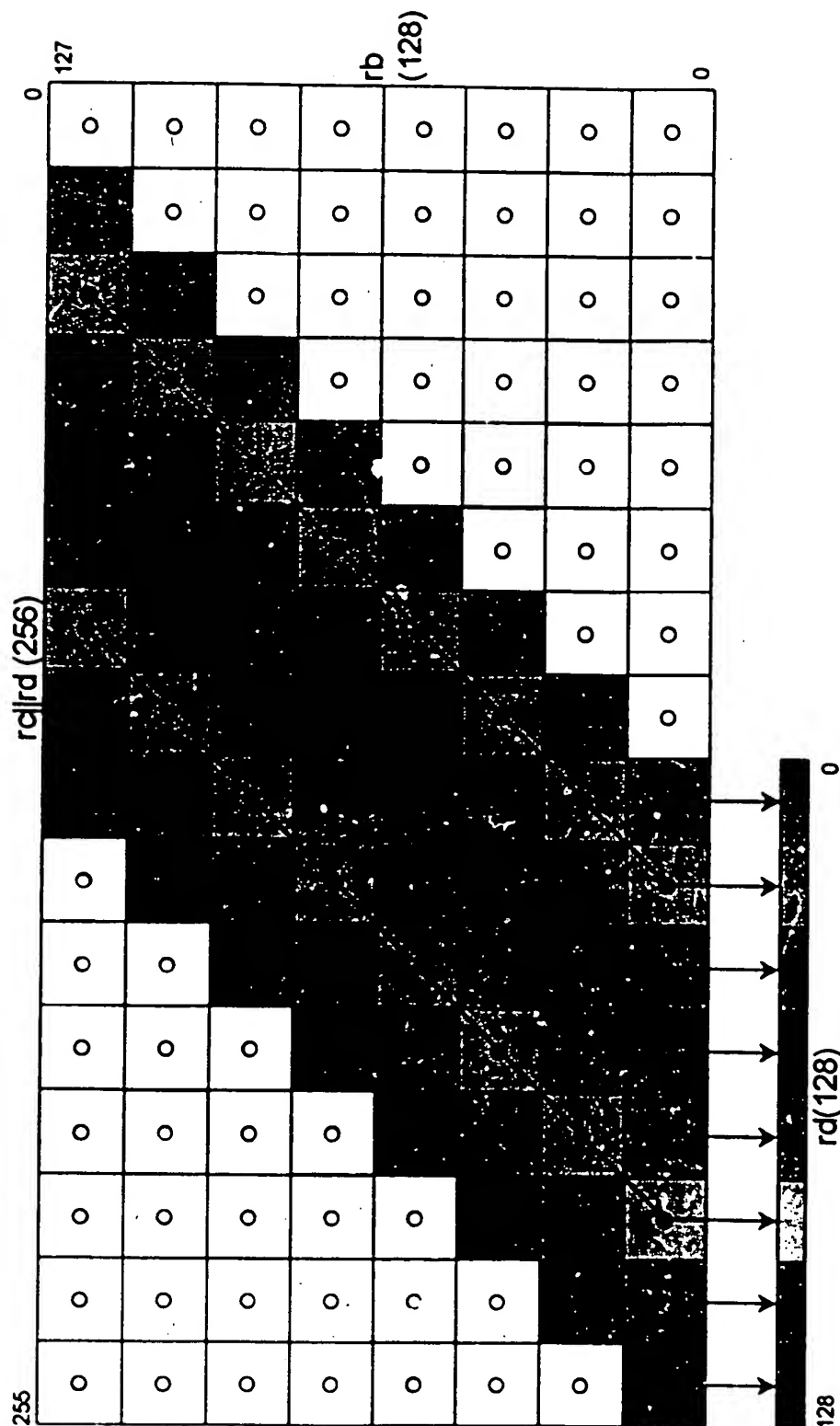
$$\mathbf{rd}_{128} = (\mathbf{rc} \parallel \mathbf{rd})_{256} * \mathbf{rb}_{128}$$





Ensemble convolve floating-point

$$\blacksquare \text{rd}_{128} = (\text{rc} \parallel \text{rd})_{256} * \text{rb}_{128}$$



Ensemble convolve complex floating-point

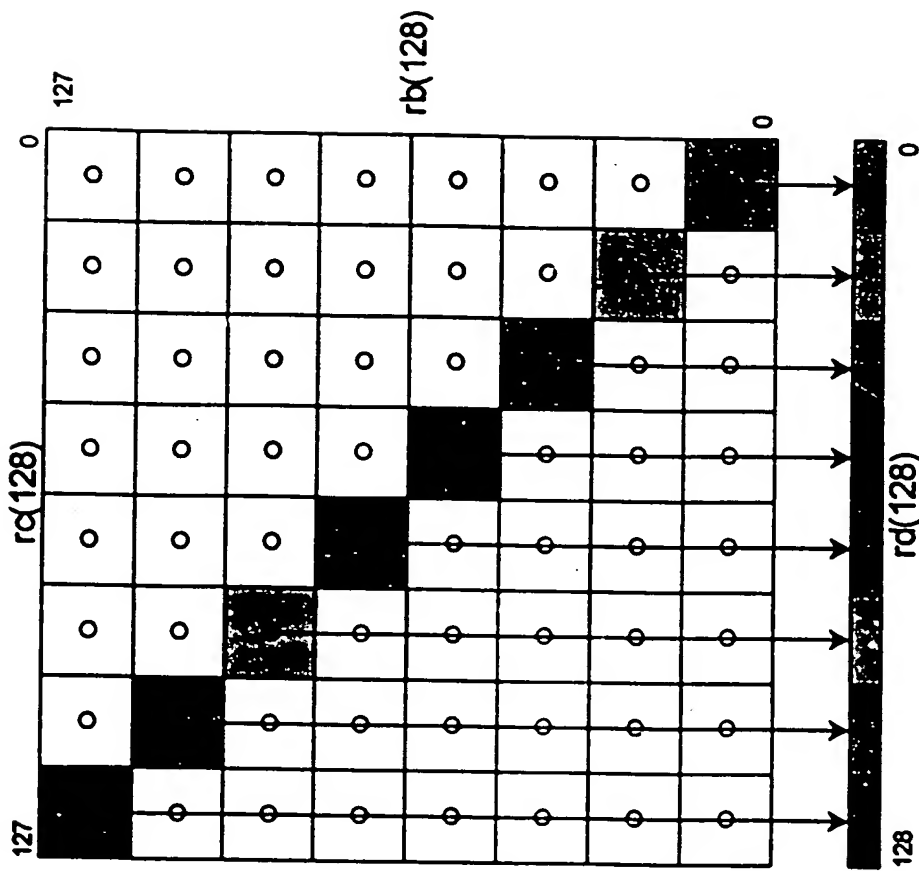
$$\blacksquare \text{rd}_{128} = (\text{rc} \parallel \text{rd})_{256} * \text{rb}_{128}$$





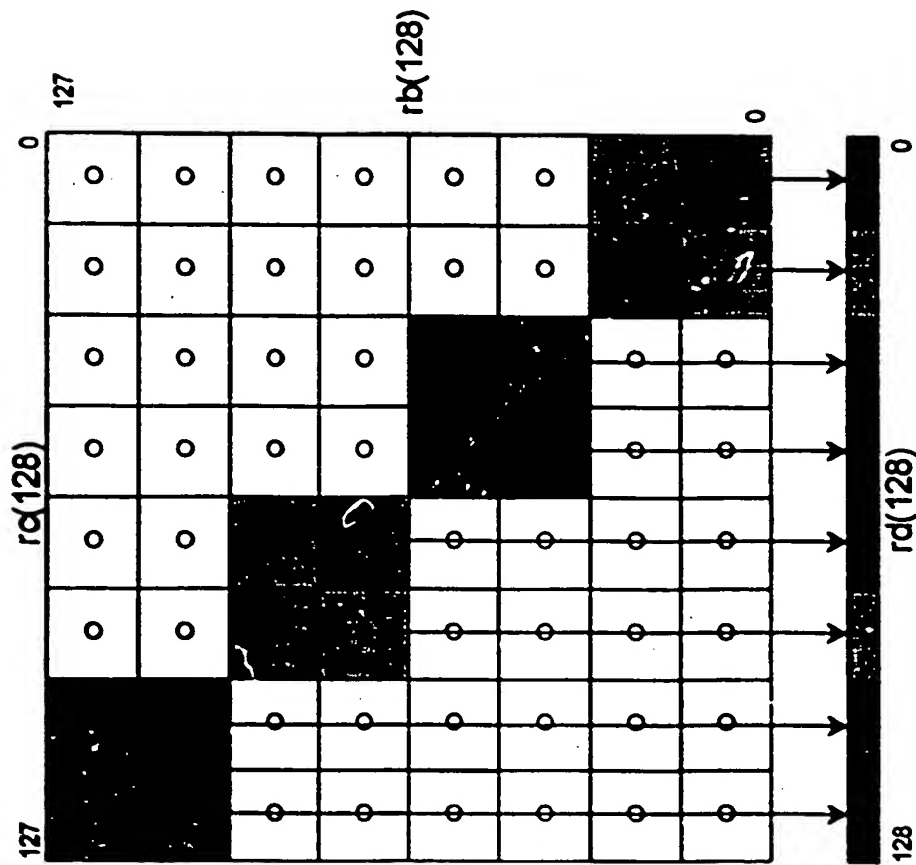
Ensemble multiply floating-point

$$\blacksquare rd_{128} = rc_{128} * rb_{128}$$



Ensemble multiply floating-point complex

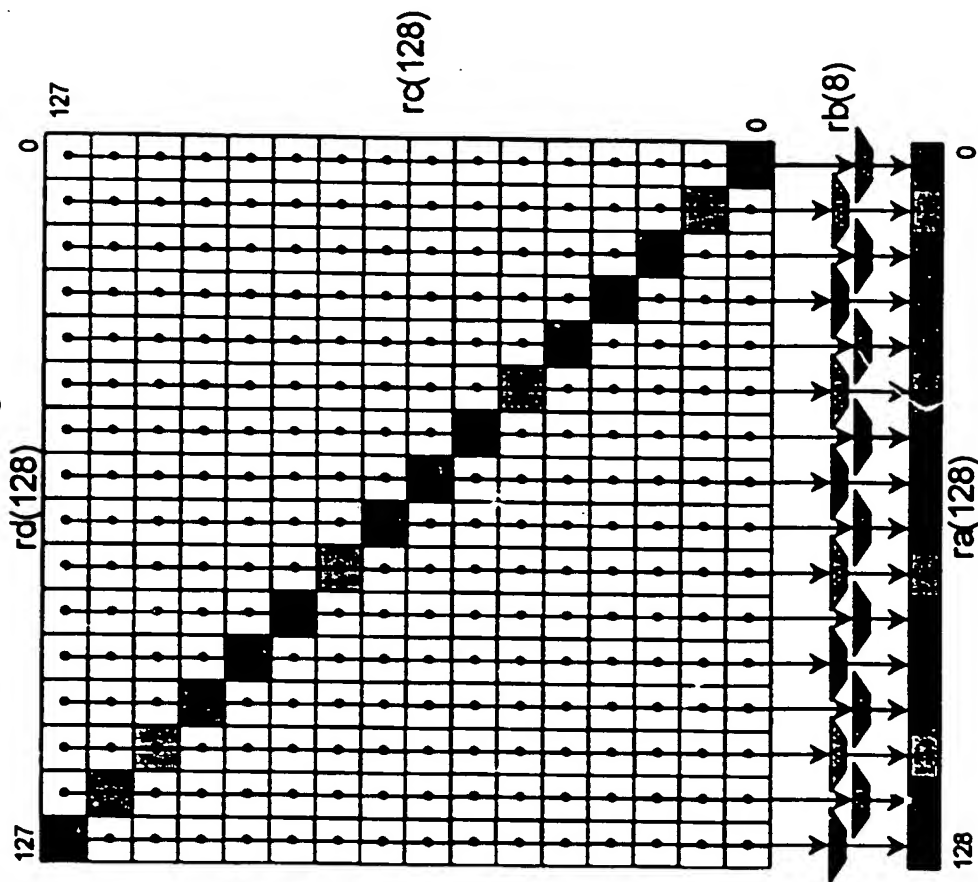
$$\blacksquare \text{rd}_{128} = \text{rc}_{128} * \text{rb}_{128}$$





Ensemble multiply Galois

$$\blacksquare ra_{128} = rd_{128} * rc_{128} \bmod rb_8$$





Wide Instructions

- Wide Multiply Matrix E
- Wide Switch X
- Wide Table L



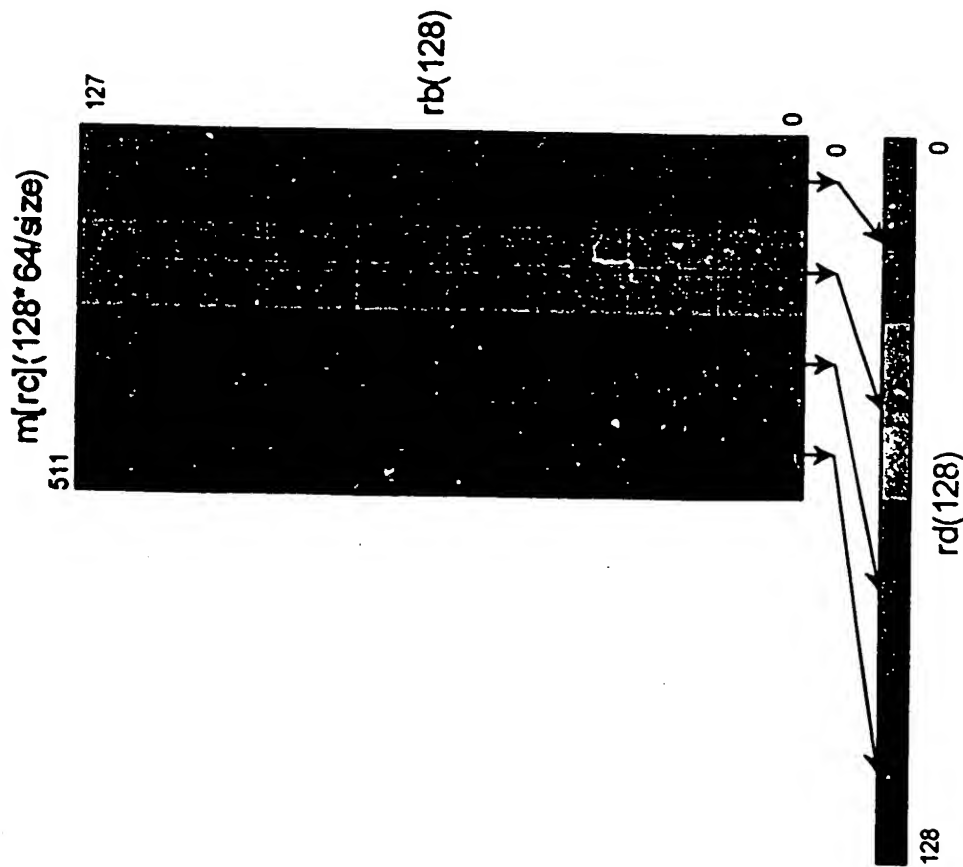
Wide size and shape

- operations up to 128x128
- full size not always required
- optional bits set in address
 - ◆ sets operand size
 - ◆ sets operand width
- operand aligned to specified size
- smaller size may use fewer cycles
 - ◆ to load operand cache
 - ◆ to perform operation



Wide multiply matrix

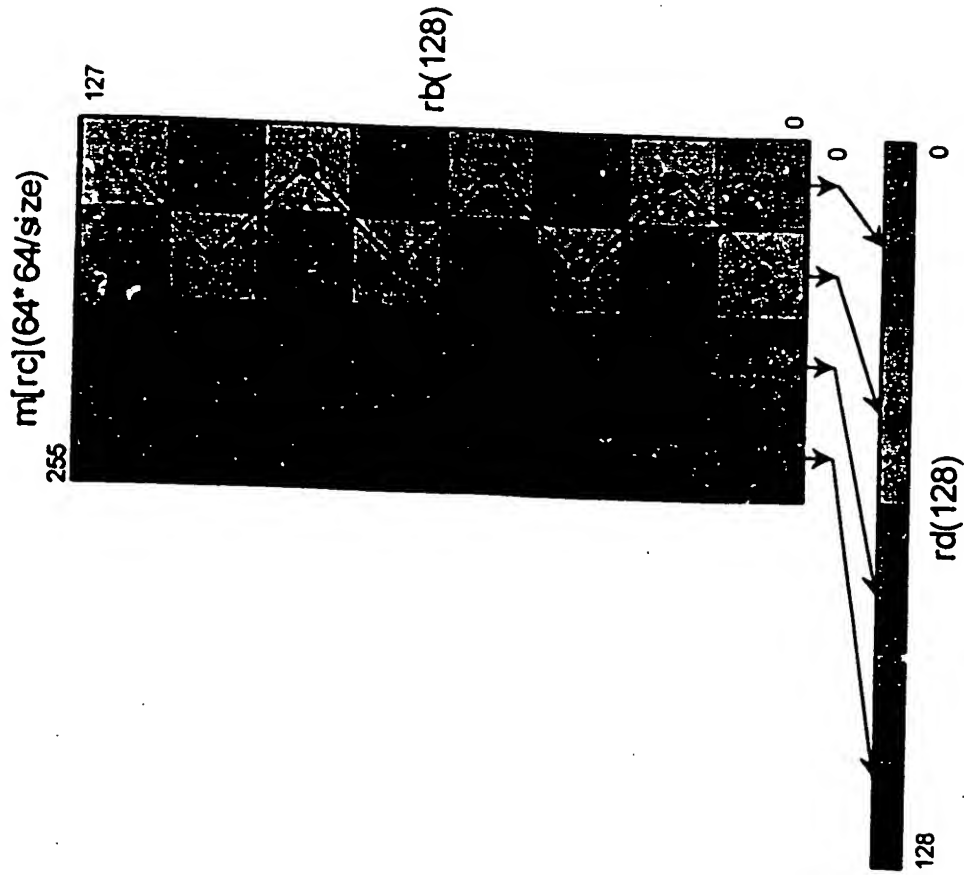
$$\blacksquare rd_{128} = m[rc]_{(128 \times 64 / size)} * rb_{128}$$





Wide multiply matrix complex

$$\blacksquare \text{rd}_{128} = \text{m}[\text{rc}]_{(64*64/\text{size})} * \text{rb}_{128}$$

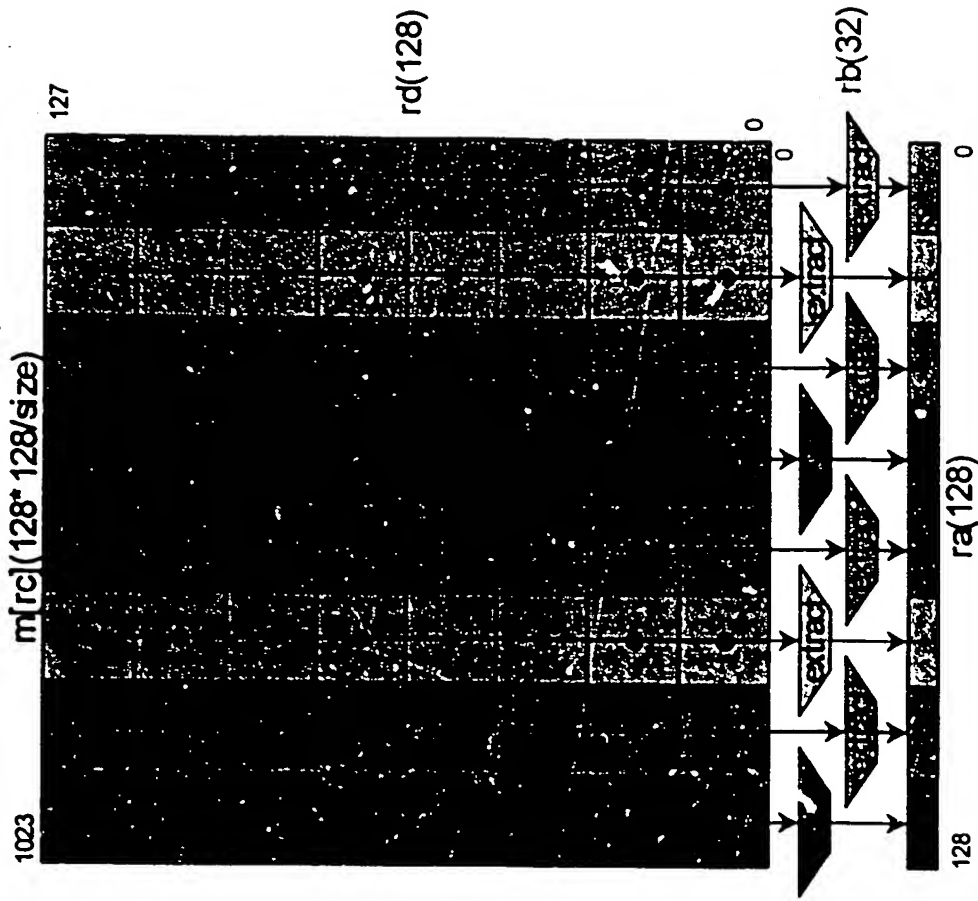


August 20, 1999



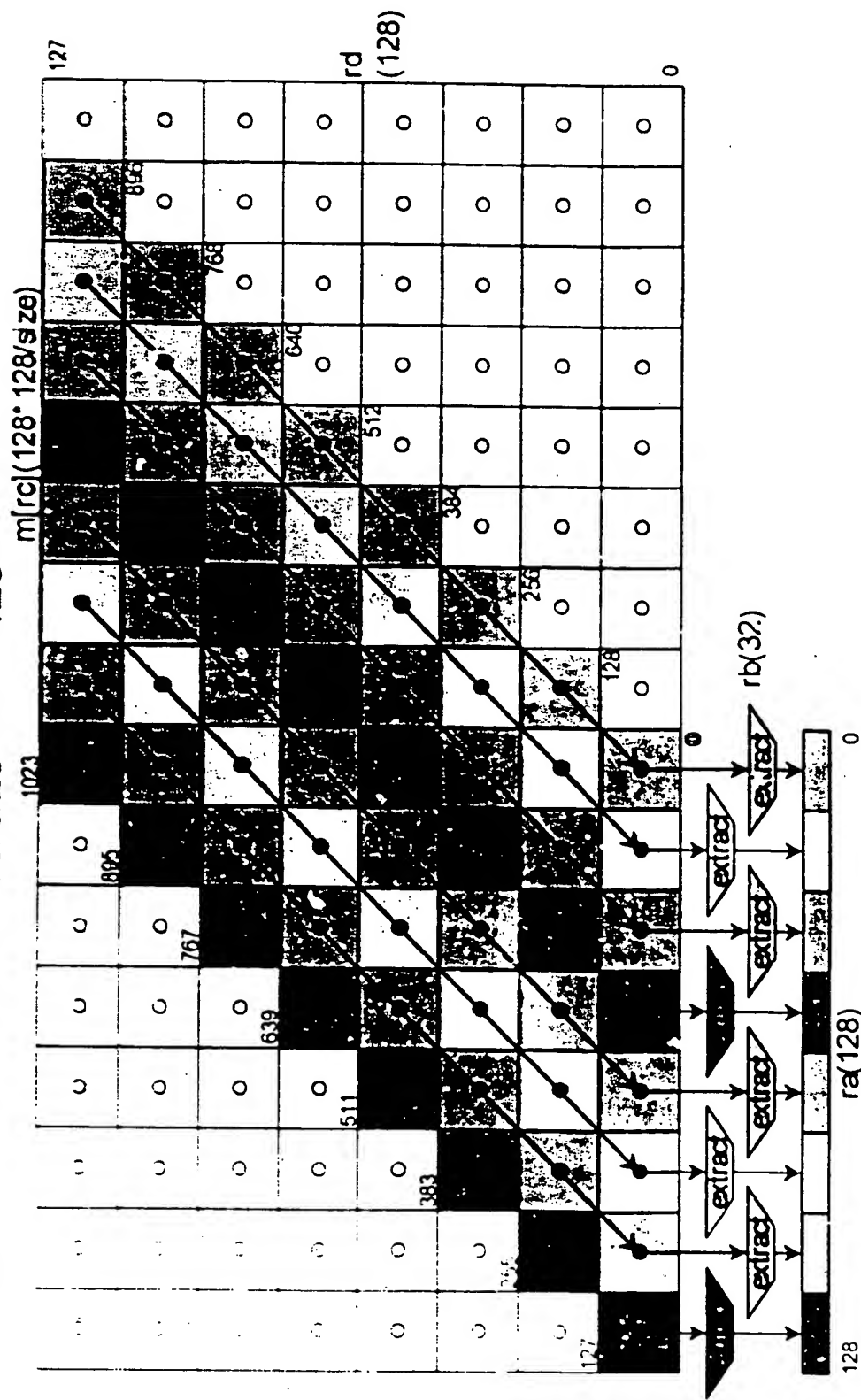
Wide multiply matrix extract

$$\blacksquare \text{ra}_{128} = \text{m}[\text{rc}]_{128 \times 128 / \text{size}} * \text{rd}_{128}$$



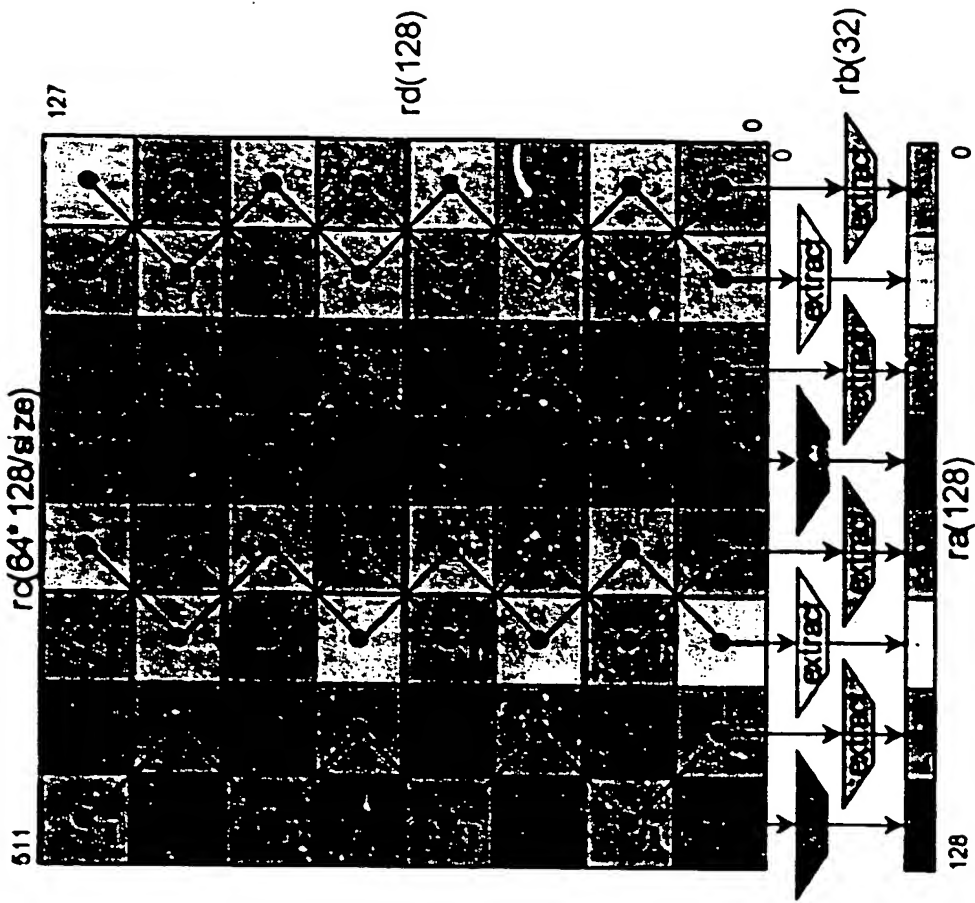
Wide multiply matrix extract

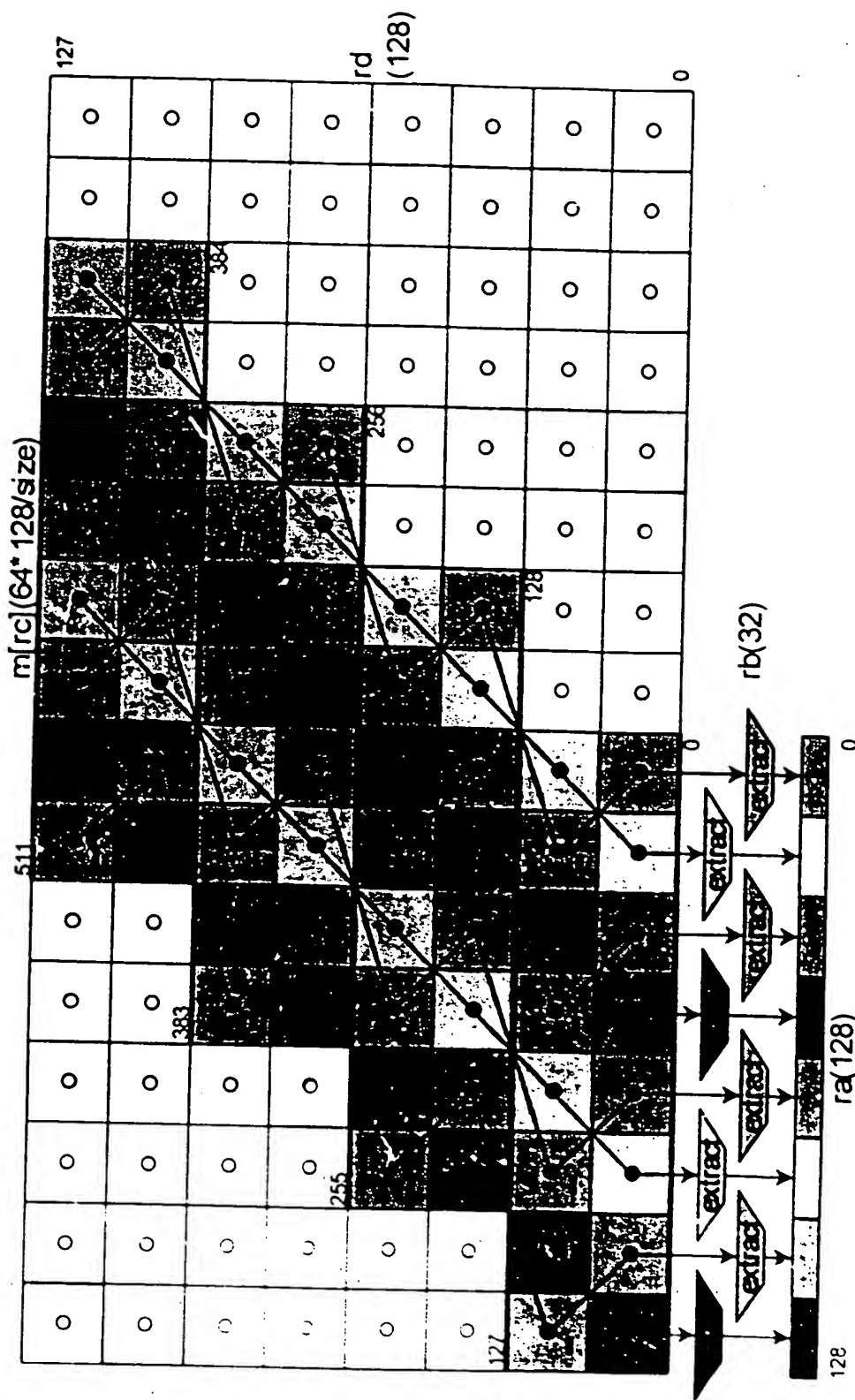
$$\blacksquare \text{ra}_{128} = \text{m}[\text{rc}]_{128 \times 128 / \text{size}} * \text{rd}_{128}$$



Wide multiply matrix extract complex

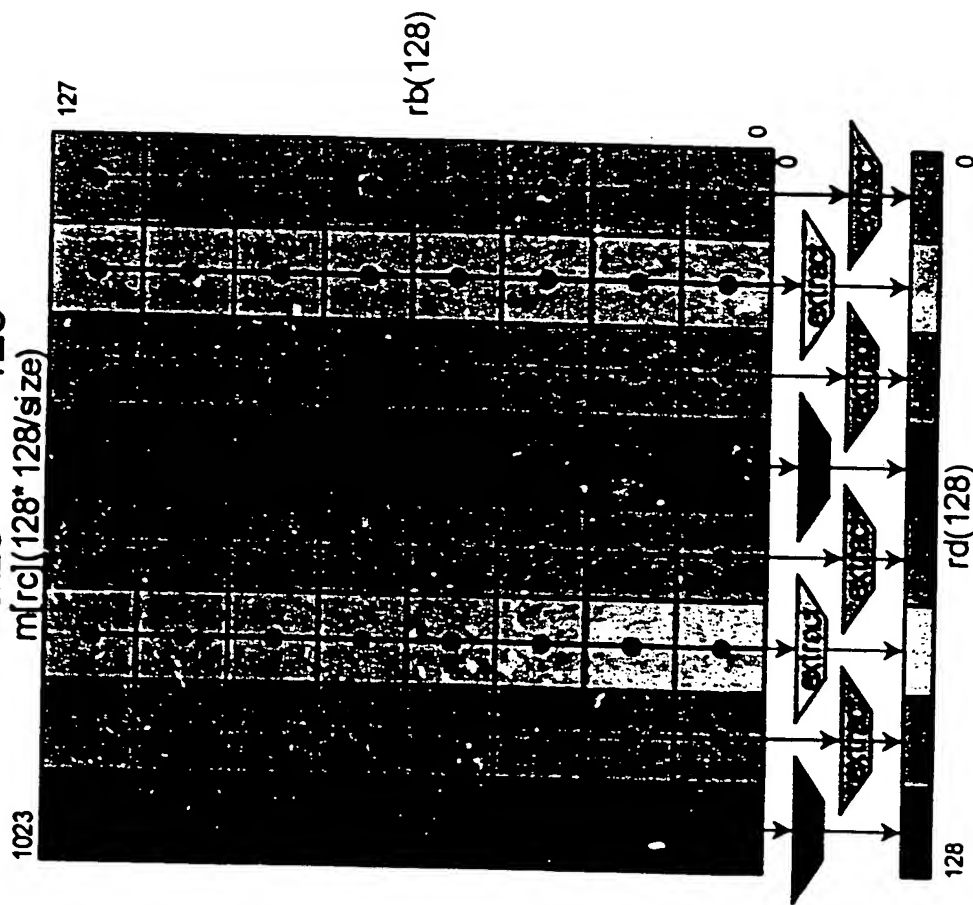
$$\blacksquare ra_{128} = m[rc]_{64 \times 128 / size} * rd_{128}$$



$$\blacksquare \text{ ra}_{128} = \text{m}[\text{rc}]_{64 \times 128/\text{size}}^* \text{ rd}_{128}$$


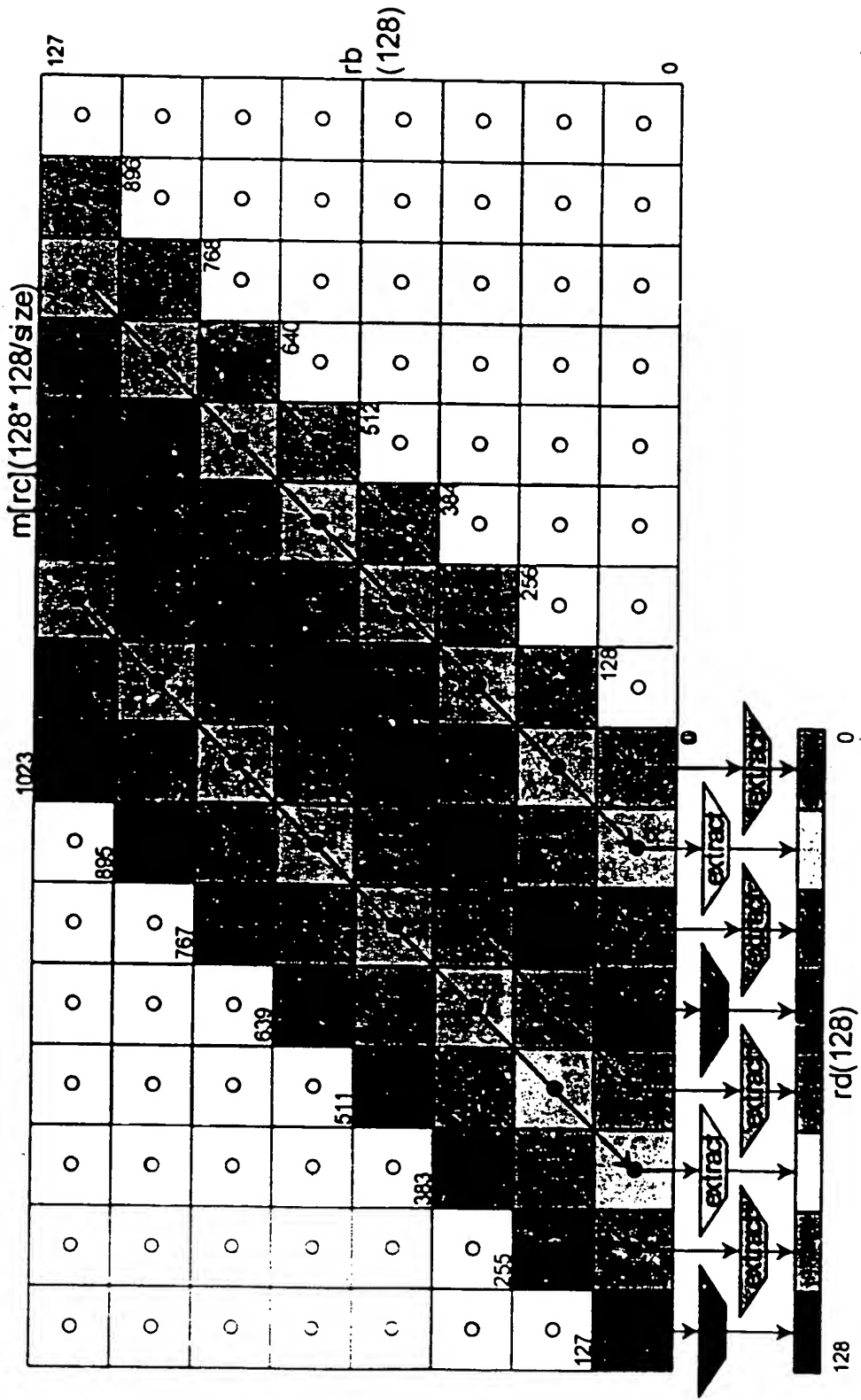
Wide multiply matrix extract immediate

$$\blacksquare \text{rd}_{128} = \text{m}[\text{rc}]_{128 \times 128 / \text{size}} * \text{rb}_{128}$$



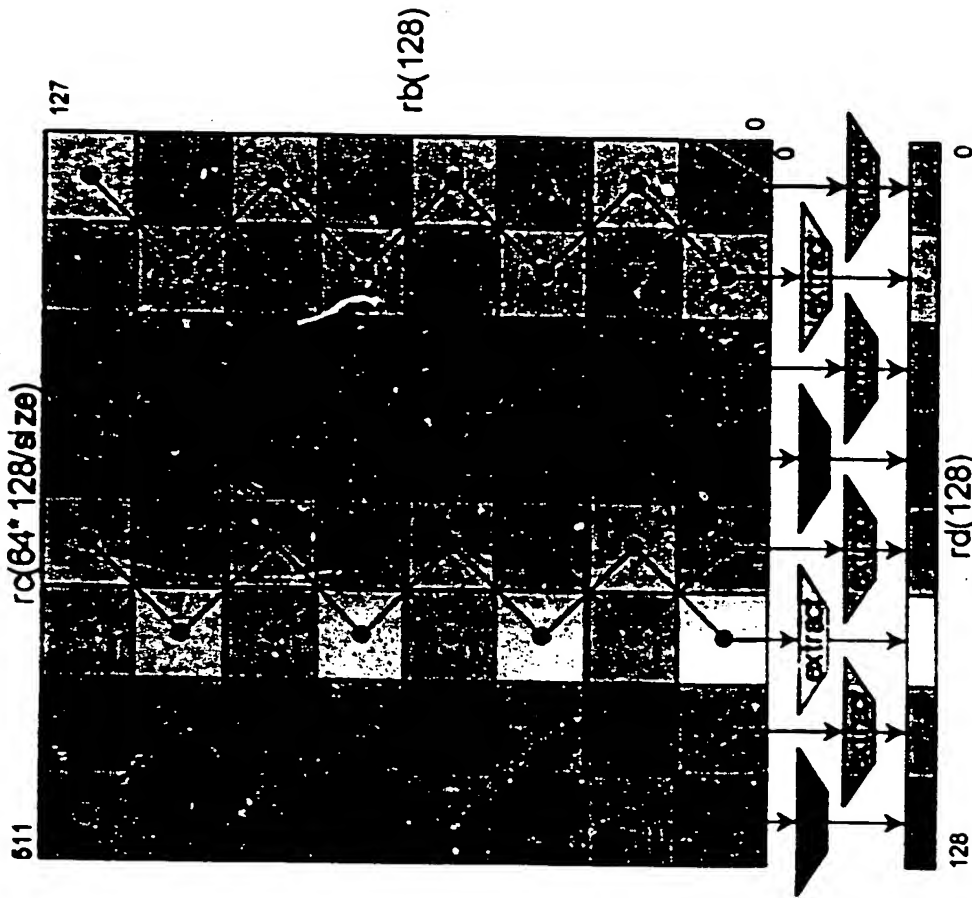
Wide multiply matrix extract immediate

$$\blacksquare \text{rd}_{128} = \text{m}[\text{rc}]_{128 \times 128 / \text{size}} * \text{rb}_{128}$$



Wide multiply matrix extract immediate complex

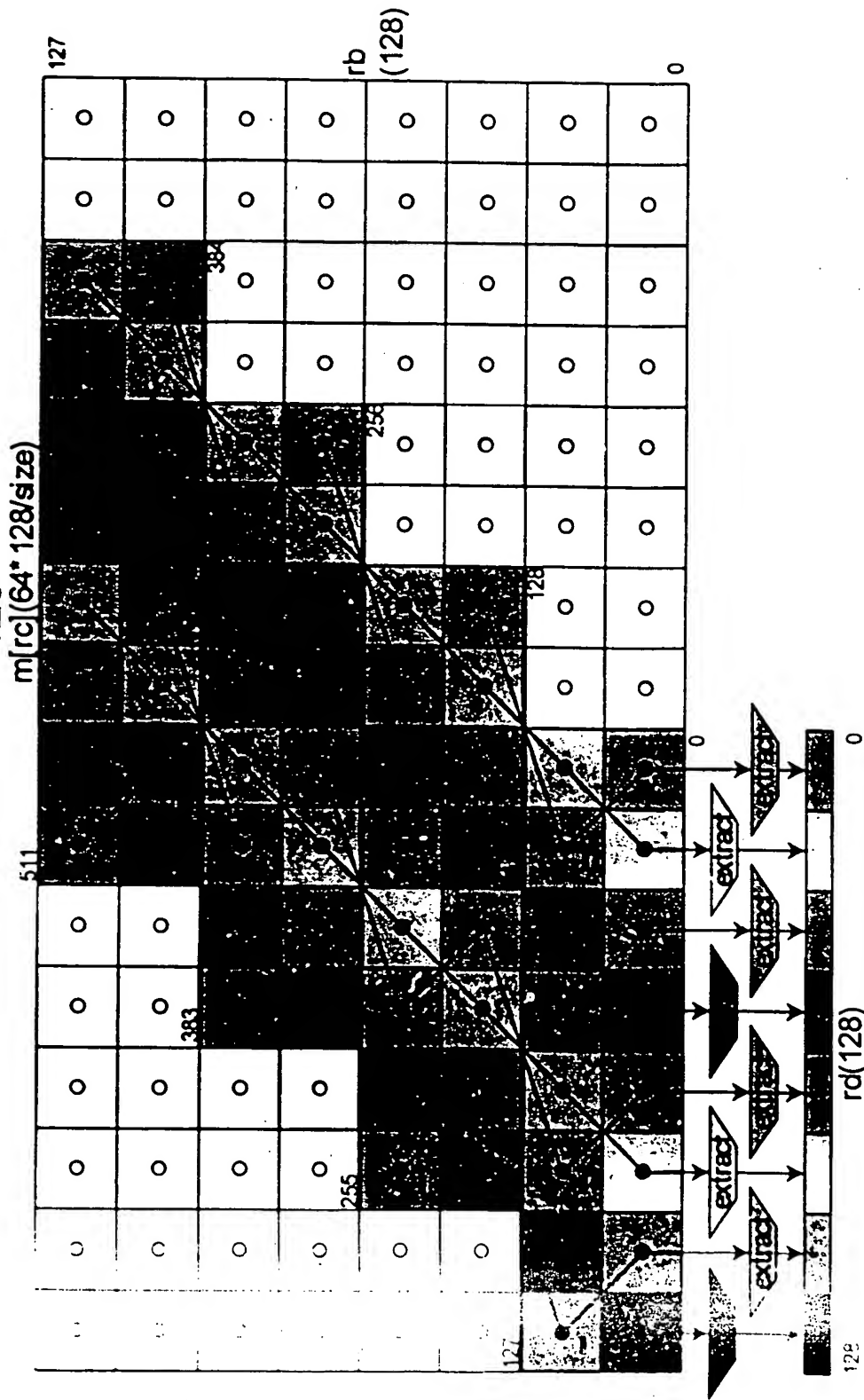
$$rc_{128} = m[rc]_{64 \times 128 / \text{size}} * rb_{128}$$





Wide multiply extract immediate complex

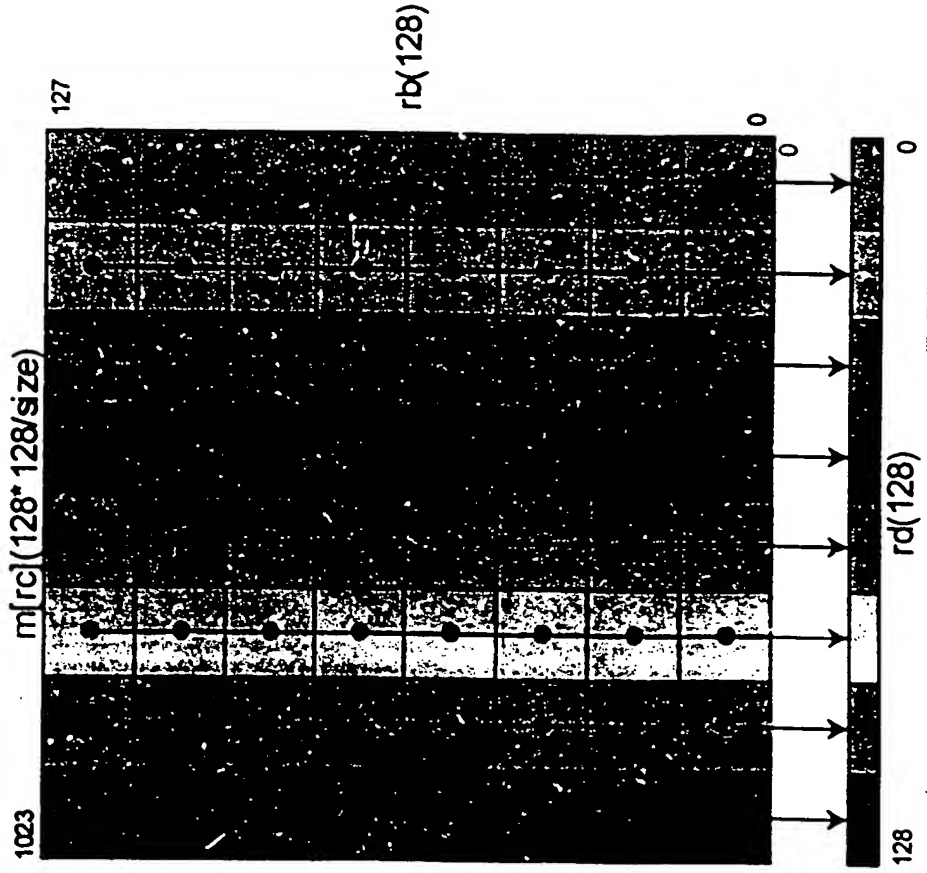
$$\blacksquare \text{rd}_{128} = m[\text{rc}]_{64 \times 128 / \text{size}} * \text{rb}_{128}$$





Wide multiply matrix floating-point

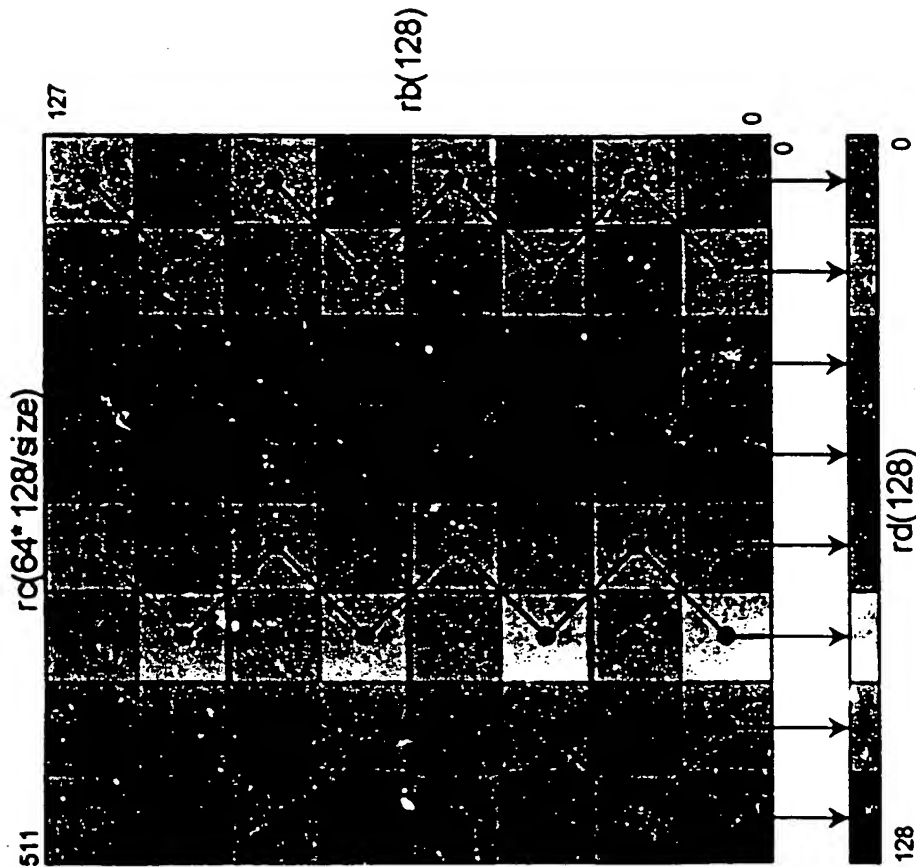
$$\blacksquare \text{rd}_{128} = \text{m}[\text{rc}]_{128 * 128 / \text{size}} * \text{rb}_{128}$$





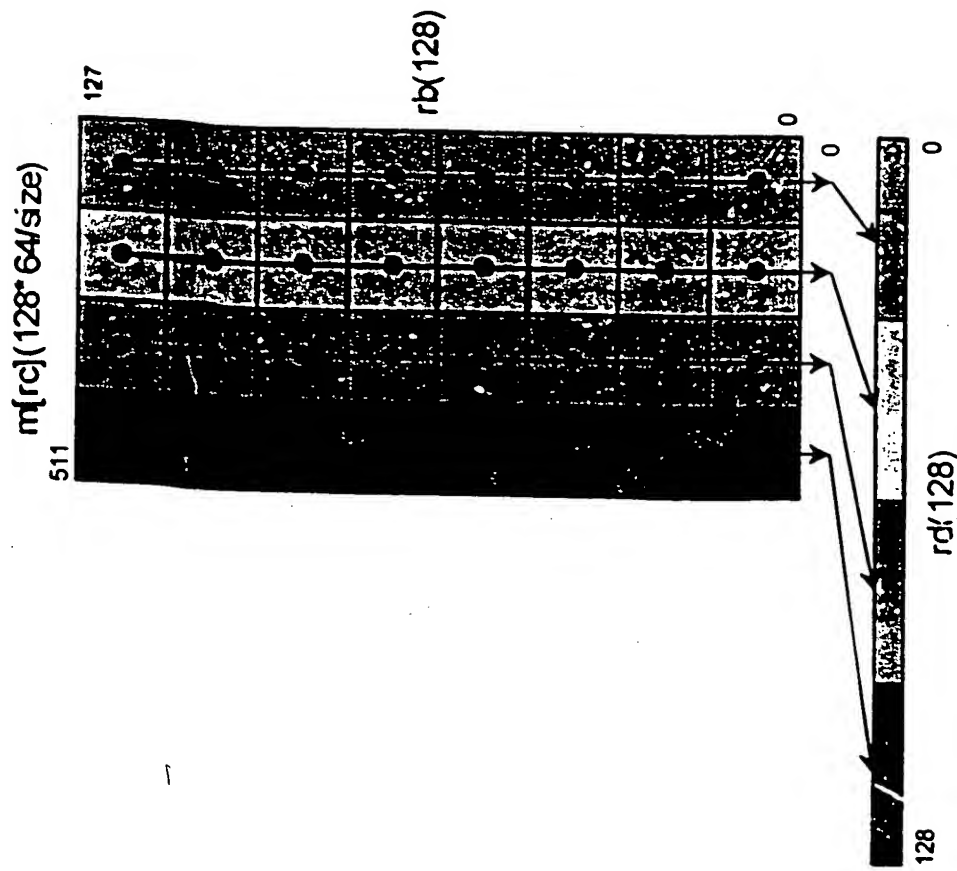
Wide multiply matrix complex floating-point

$$\blacksquare \text{rd}_{128} = \text{m}[\text{rc}]_{64 \times 128/\text{size}} * \text{rb}_{128}$$



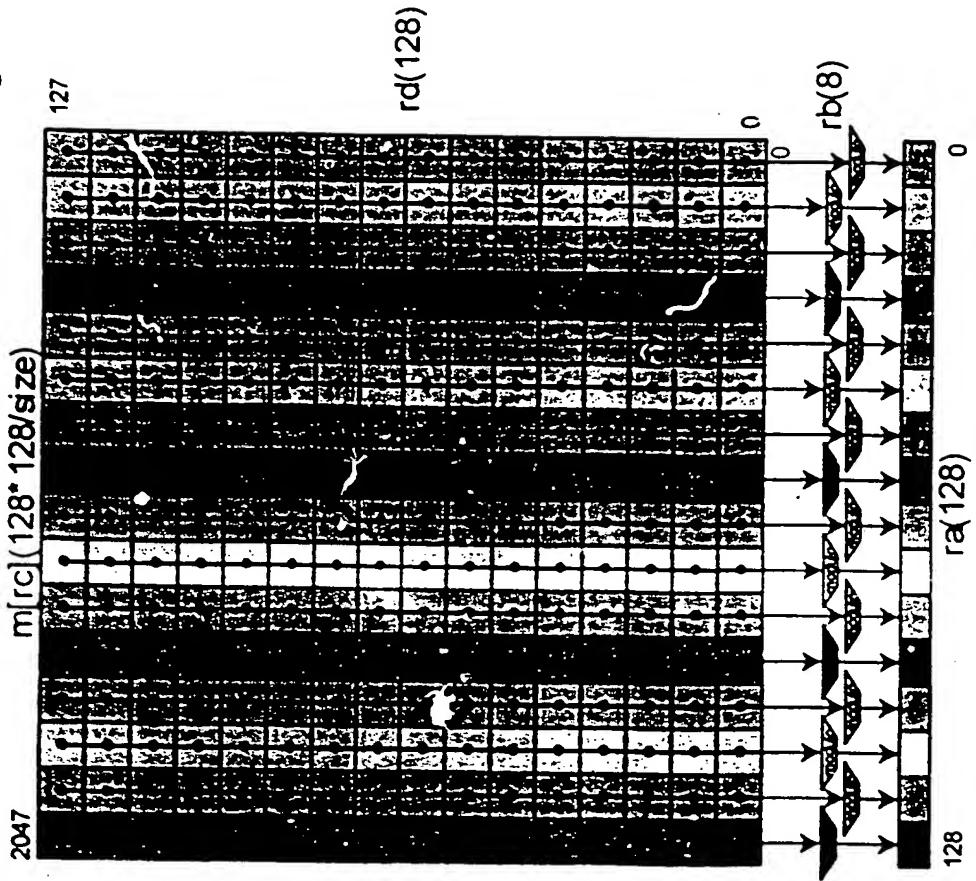
Wide multiply matrix polynomial

$$\blacksquare \text{rd}_{128} = \text{m}[\text{rc}]_{(128 \times 64/\text{size})} * \text{rb}_{128}$$



Wide multiply matrix Galois

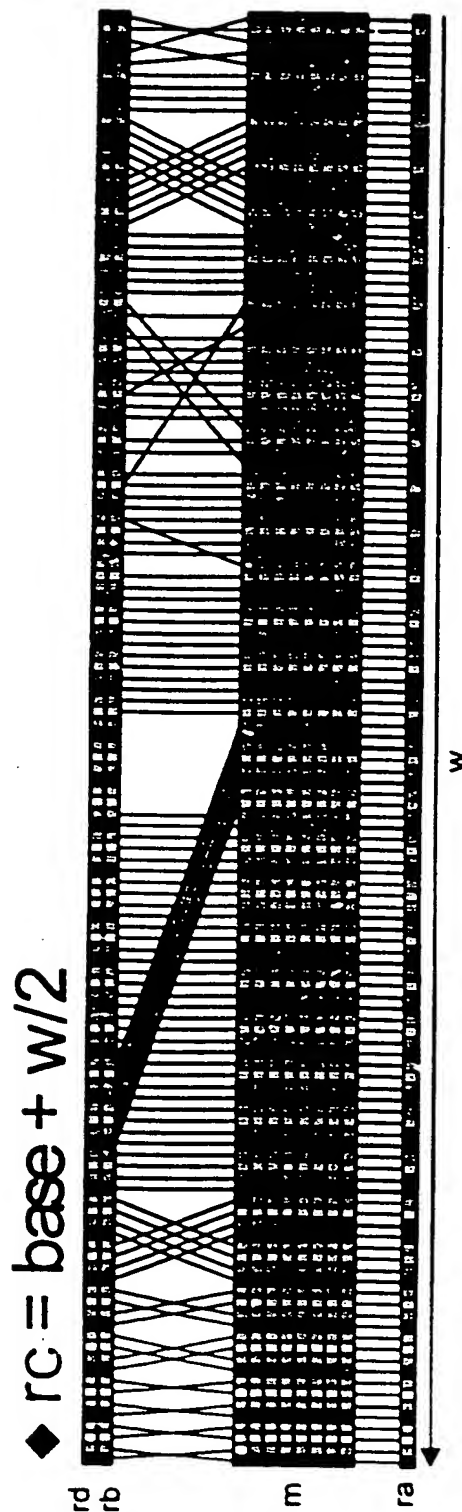
$$\blacksquare ra_{128} = m[rc]_{128 \times 128 / size} * rd_{128} \bmod rb_8$$





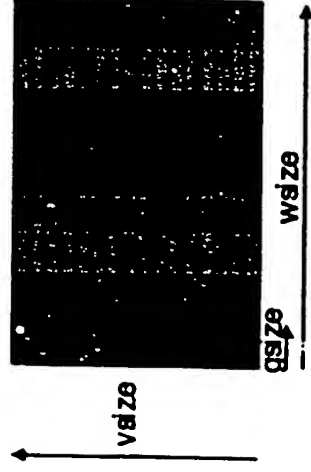
Wide switch

- $j(i) = m[rc]_{7w+i, 6w+i, 5w+i, 4w+i, 3w+i, 2w+i, w+i, i}$
- $ra_i = (rd \parallel rb)_j, i=0..127$
- rc specifies address and w



Wide Table

- Table lookup
 - ◆ msiz: total table size
 - ◆ wsize: table width
 - ◆ vsize: table depth
 - ◆ gsize: Group size (table granularity)
- $j(i) = b_{\lfloor vsize-1+i \rfloor * wsize + i \rfloor wsize - 1}..0$
- $rd_{i+gsiz-1..i} = m[rc]_{j+gsiz-1..j}, i=0..128-gsiz \text{ by } gsize$
- rc specifies address, msiz, wsize
 - ◆ $rc = base + msiz/16 + wsize/16$
 - ◆ $vsize = msiz/wsize$



Summary

- Order-of-magnitude multiply performance increase
 - ◆ matrix multiply
 - ◆ convolve
- Wide switch: bit permutation
- Wide select: table lookup



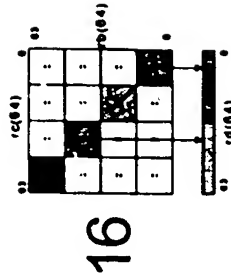
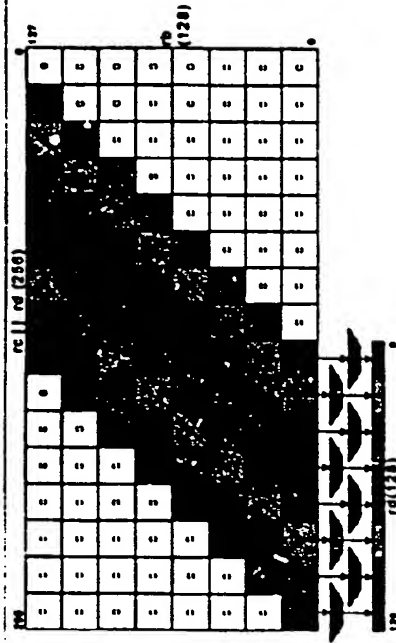
BroadMX™ vs. MMX™

■ Convolve Extract

- ◆ 64 Multiplies
- ◆ 56 Adds
- ◆ 08 Extract w/round

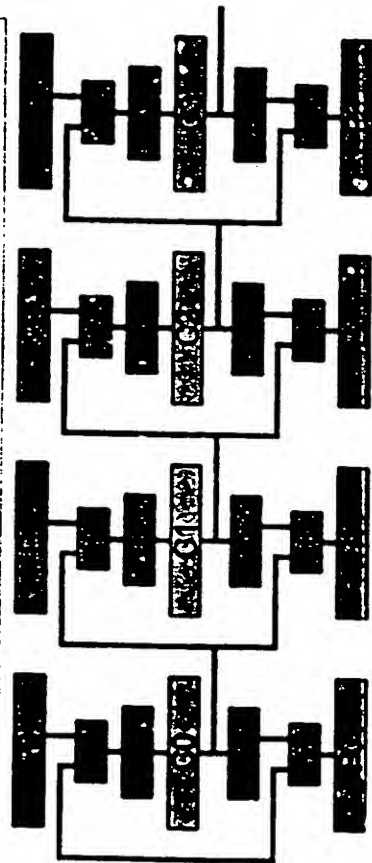
■ MMX Instructions

- ◆ 16 MOV
 - ◆ 16 PMADDWD
 - ◆ 12 PADDD
 - ◆ 08 PSHW
 - ◆ 04 PSHR
 - ◆ 02 PACK
-
- ◆ 58 total



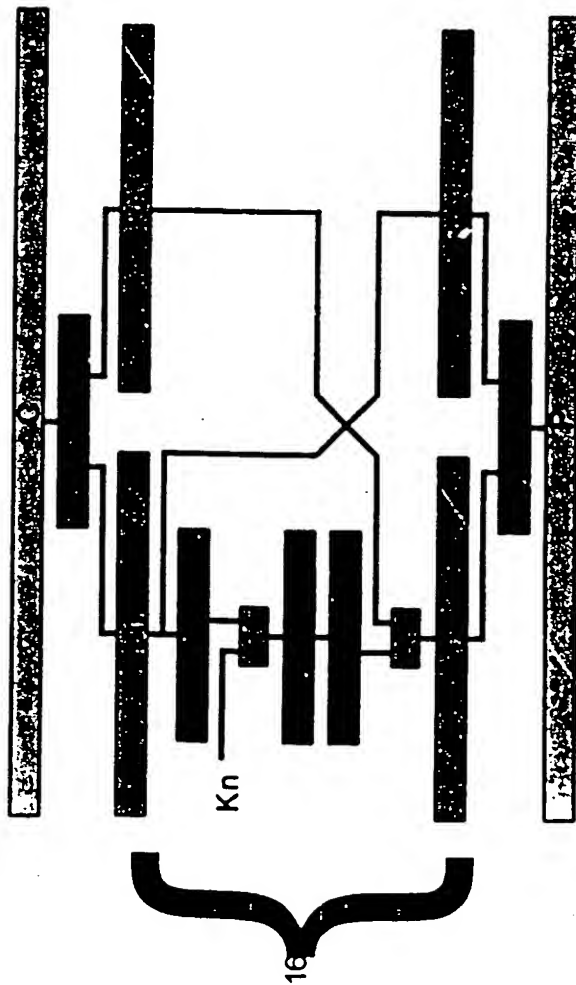
DES decryption

- CBC (Cypher Block Chaining) decrypt uses parallelism between blocks



- DES decrypt

- ◆ E expansion
- ◆ + key xor
- ◆ S substitution
- ◆ P permutation
- ◆ + data xor



Software DES

■ Optimizations

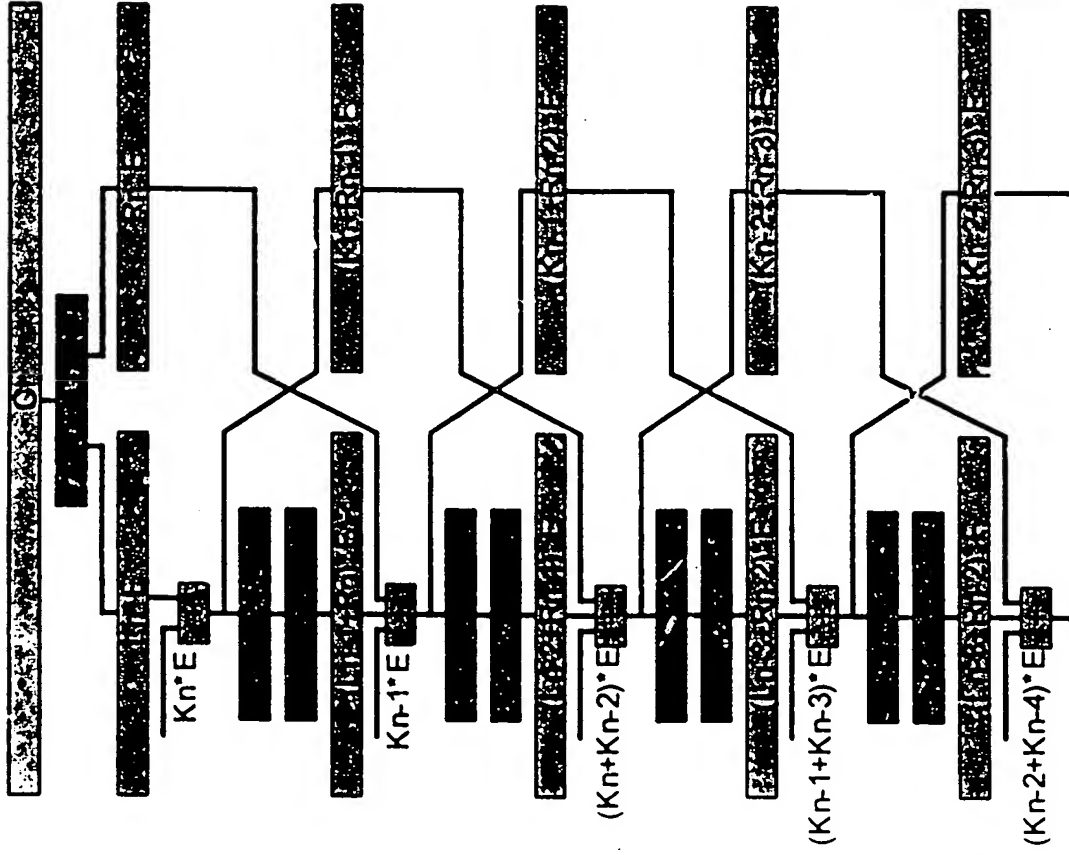
- ◆ 2 blocks/register
- ◆ 4 blocks at once
- ◆ distribute E
- ◆ combine + +

■ Code

K, + L.128, G.XXX
S W.TRANSLATE
PE W.SWITCH

■ Performance

- ◆ 52 cycles/4 blocks
- ◆ 985Mbps@200MHz
- ◆ 10x per clock over fastest sw DES

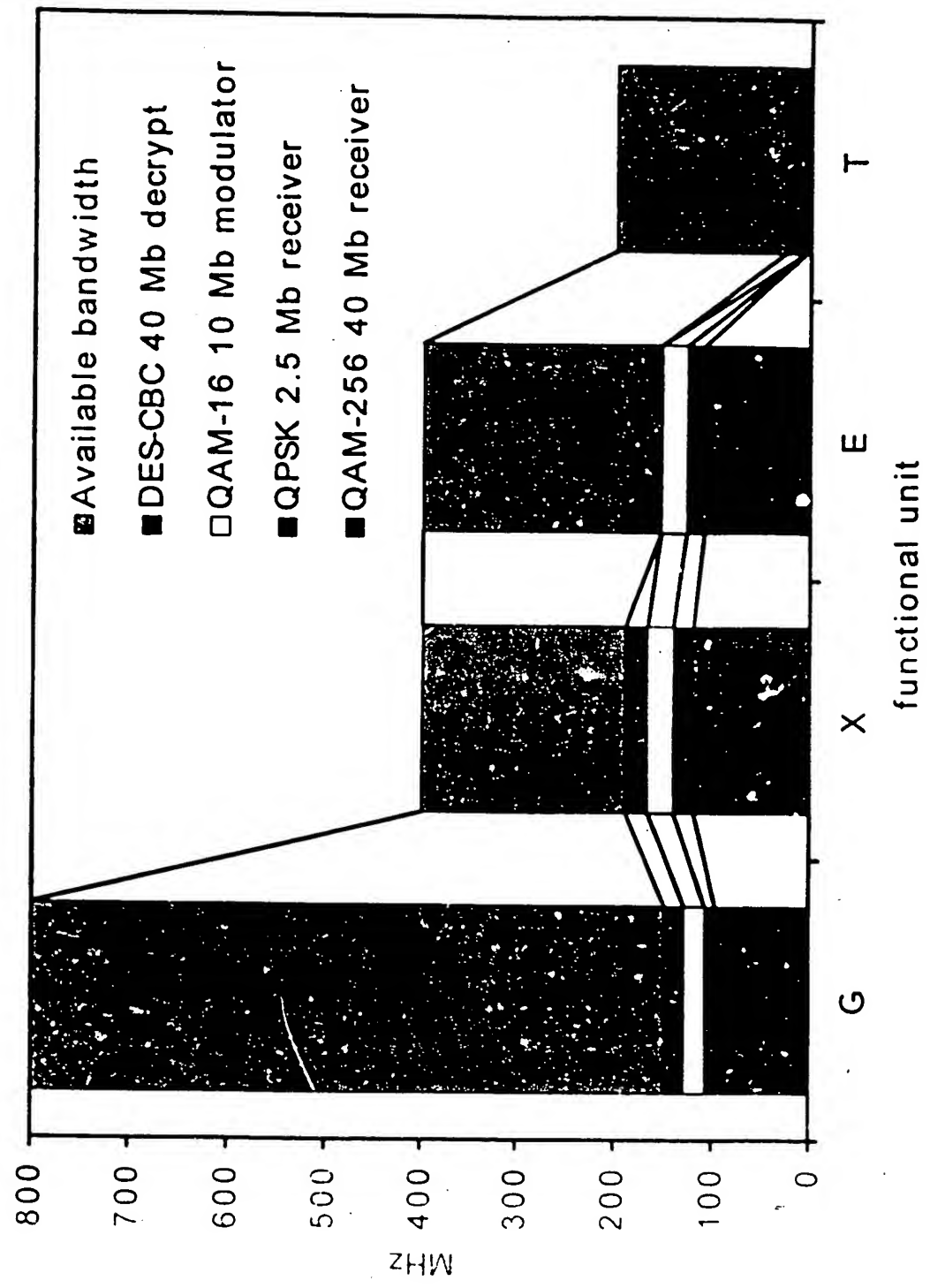


Software DES

- DES standard at end of 20 year life
 - ◆ brute-force code-breaking
 - \$10000 RSA DES Challenge
 - Electronic Frontier Foundation (EFF)
 - ◆ 56 hours to crack
 - ◆ \$200k to design and build
 - ◆ FIPS standard expire this year
- Handles DES extensions
 - ◆ larger keys, bigger S-boxes
 - ◆ more rounds, larger blocks
 - ◆ soft S-boxes and P-boxes
- AES standard in development
 - ◆ 15 official candidates
 - ◆ new standard unpredictable



Instruction bandwidth for cable modem



Software tools

■ Compiler-based development tools

♦ C, C++ compiler

- intrinsic functions, function inlining
- register allocation, code scheduling
- future: automatic parallelisation

♦ object-module tools

- linker, libraries, debugger

■ OS: RT microkernel, Linux

■ DSP libraries

■ Sophisticated tools

- Mathematica: symbolic verification
- GOPS: cross-development library